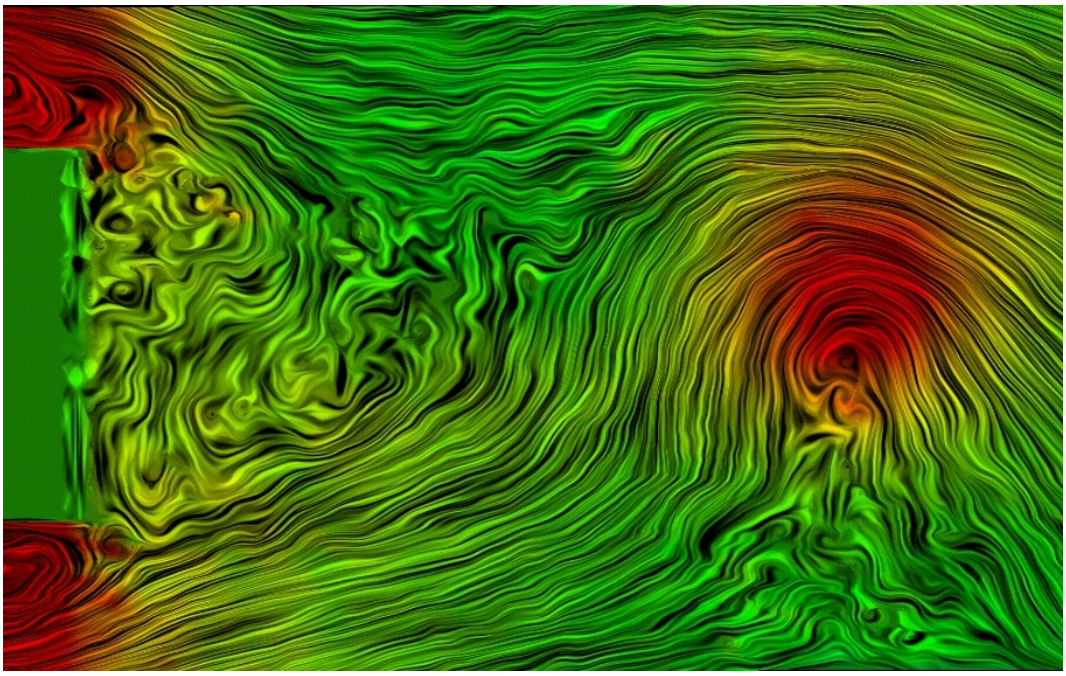
university of groningen

faculty of mathematics and natural sciences

COMPUTATIONAL FLUID DYNAMICS

A.E.P. Veldman



Lecture Notes in Applied Mathematics

Academic year 2012–2013

COMPUTATIONAL FLUID DYNAMICS

Code: WICFD-03

MSc Applied Mathematics
MSc Applied Physics
MSc Mathematics
MSc Physics

Lecturer: A.E.P. Veldman
University of Groningen
Institute for Mathematics and Computer Science
P.O. Box 407
9700 AK Groningen
The Netherlands

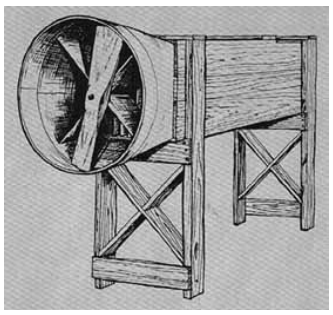
The cover picture is a snapshot from a three-dimensional simulation of turbulent flow past a rectangular cylinder at a Reynolds number of 22,000. The simulation has been carried out by Roel Verstappen (RUG) using a symmetry-preserving discretization method. The visualization of the (instantaneous) streamlines has been carried out by Wim de Leeuw (CWI) using a spot-noise technique. For more details we refer to Section 3.4.1.

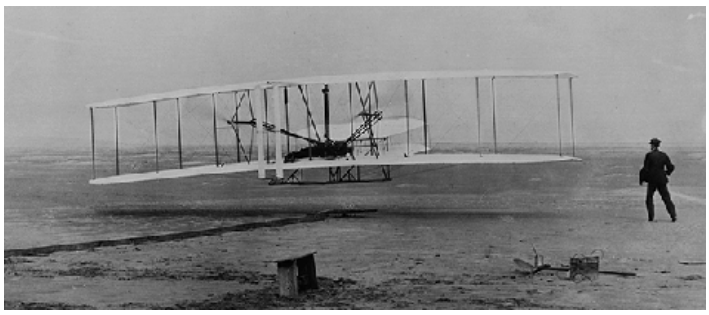
Preface

There is fluid flow everywhere around us: air is flowing past airplanes, cars and buildings; water is flowing past ships, through rivers and harbours; oil is flowing through pipelines and in underground reservoirs; blood is flowing through our arteries. And the most important flow around us has not even been mentioned: the air flow in the atmosphere that determines our weather.

It will be clear that for weather prediction and for the design of airplanes, cars, ships, etc. knowledge of flow phenomena is essential. Such knowledge can be obtained along three ways: experiment, theory and computer simulation.

Experiment: The oldest way of acquiring flow knowledge is by experiment. The Wright brothers already had built a small windtunnel to design their first airplane. Currently windtunnels are usually impressive buildings; the experiment time required for a new design is typically 10,000 to 20,000 hours. Even with two shifts a day (i.e. 16 working hours per day) such an experiment program takes three to five years. And the time for making the scale model, and for analysing the measurement data has not even been accounted for. Today, such a long time for development is not acceptable anymore, and a faster way has to be found.





Theory: Flow research can also be carried out along a theoretical way. More than one and a half century ago already (Navier 1823, Stokes 1845) the equations describing the flow of air and water were derived: the Navier–Stokes equations. With pencil and paper these equations cannot be solved. Only if the equations are simplified strongly, this theoretical approach can produce adequate information.

Computer simulation: In the modern computer era, another approach has become feasible: computer simulation. Here, the Navier–Stokes equations are solved with methods developed in the realm of numerical mathematics. There is still a role for experiments, but different from before: experiments will mainly serve as validation of the computational results.

Groningen, January 2014.

Contents

Chapter 1

The convection-diffusion equation

1.1 Analytical formulation

The discussion of numerical solution methods for the equations of fluid flow starts with a 'simple' situation: the transport of a solute in a flowing medium. Two transport mechanisms can be distinguished: convection[1] and diffusion:

convection	=	transport due to the motion of the medium;
diffusion	=	transport due to differences in concentration.

The concentration of solute is denoted by $\phi(\boldsymbol{x}, t)$, where $\boldsymbol{x}$ represents space and t time. The flowing medium is assumed to be incompressible, with a velocity $\boldsymbol{u}(\boldsymbol{x}, t)$.

The equation describing the concentration as a function of space and time can be derived from a conservation law. Hereto, consider an arbitrary volume Ω with boundary Γ and outward pointing normal $\boldsymbol{n}$. A decrease of the amount of solute inside the volume Ω is due to outward transport of solute through the boundary Γ. The convective transport per unit of time is given by

$$\int_\Gamma \phi\, \boldsymbol{u} \cdot \boldsymbol{n}\, \mathrm{d}\Gamma;$$

the diffusive transport by

$$\int_\Gamma -k\ \mathrm{grad}\, \phi \cdot \boldsymbol{n}\, \mathrm{d}\Gamma,$$

where the diffusion has been taken proportional to the gradient of the concentration (diffusion coefficient $k \geq 0$).

Conservation of mass in Ω yields

$$\begin{aligned} \int_\Omega \frac{\partial \phi}{\partial t}\, \mathrm{d}\Omega &= -\int_\Gamma (\phi\, \boldsymbol{u} - k\ \mathrm{grad}\, \phi) \cdot \boldsymbol{n}\, \mathrm{d}\Gamma = (\text{Gauss}) \\ &= -\int_\Omega \mathrm{div}\, (\phi\, \boldsymbol{u} - k\ \mathrm{grad}\, \phi)\, \mathrm{d}\Omega. \end{aligned}$$

[1] Strictly spoken, we should use the word 'advection'. Originally, the word 'convection' was restricted to flow induced by thermal effects, where first diffusion causes temperature differences which then result in buoyant flow due to density differences. However, in the CFD community the word 'convection' has taken over the more general meaning of 'advection'.

As this holds for any arbitrary volume Ω, it follows that

$$\frac{\partial \phi}{\partial t} + \text{div}\,(\phi\,\boldsymbol{u} - k\ \text{grad}\,\phi) = 0. \tag{1.1}$$

This is called the divergence form of the equation. When $\text{div}\,\boldsymbol{u} = 0$ (incompressibility condition) the convection-diffusion equation can be rewritten in its more common form

$$\frac{\partial \phi}{\partial t} + \boldsymbol{u} \cdot \text{grad}\,\phi = \text{div}\,(k\,\text{grad}\,\phi).$$

Remark 1 Equation (1.1) also describes heat transport in a flowing medium with ϕ the temperature. The diffusion corresponds with heat conduction.

Remark 2 The above mass balance can also be considered in discrete form; the finite-volume method results. More on this method in Section 1.6 and Appendix A.1.2.

1.2 Central versus upwind discretization

1.2.1 Analytical problem

We start with the steady convection-diffusion equation in one space dimension, with a velocity field u and a diffusion coefficient k that are taken constant

$$L\phi \equiv u\,\frac{\mathrm{d}\phi}{\mathrm{d}x} - k\,\frac{\mathrm{d}^2\phi}{\mathrm{d}x^2} = 0 \quad (0 < x < L) \quad \phi(0) = T_0, \quad \phi(L) = T_L. \tag{1.2}$$

The solution to this equation is given by

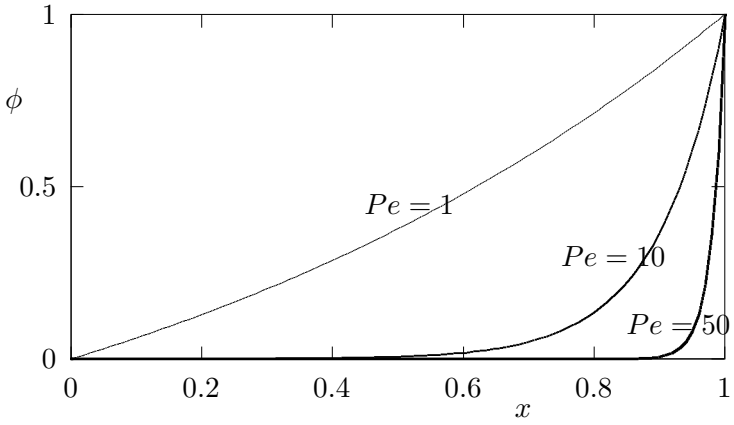
$$\phi(x) = T_0 + (T_L - T_0)\frac{1 - e^{ux/k}}{1 - e^{uL/k}} \equiv T_0 + (T_L - T_0)\frac{1 - e^{Pe(x/L)}}{1 - e^{Pe}}, \tag{1.3}$$

in which

$$Pe \equiv \frac{uL}{k} \text{ is the Péclet number.}$$

For moderate values of Pe the solution $\phi(x)$ varies smoothly over the whole domain. When Pe is large, the solution possesses a boundary-layer character in which $\phi(x) \approx T_0$, except in a thin layer of thickness $k/u(= L/Pe)$ near the outflow boundary $x = L$ where the solution adapts itself to the outflow condition $\phi(L) = T_L$.

Remark The fastest way to 'prove' that the boundary layer thickness is proportional to k/u uses dimensional analysis. First recognise that u has dimension m/sec, whereas k has dimension m^2/sec. Then conclude that k/u is the only combination that can be made with the physical parameters that has a dimension of length.



Exact solution of convection-diffusion equation for various values of the Péclet number Pe.

1.2.2 Discretization

The convection-diffusion equation (1.2) will be discretized on a grid with grid points $x_i = ih,\ h = L/I,\ i = 0, \cdots, I$. Various ways exist of discretizing partial differential equations; in these lecture notes we will follow either a finite-difference or a finite-volume approach (see Appendix A.1). Let us start simple with a finite-difference approximation of the partial derivatives occurring in (1.2).

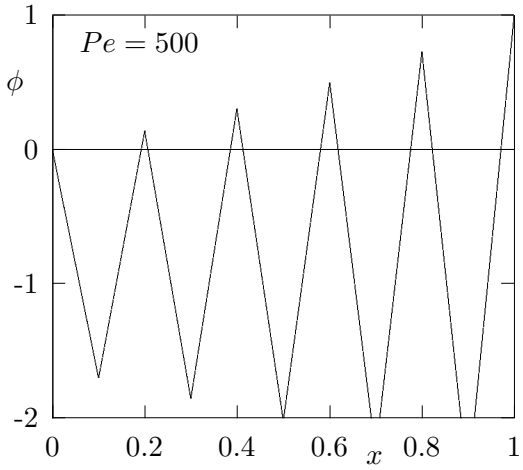
With second-order central discretization we have

$$\frac{\mathrm{d}\phi}{\mathrm{d}x} = \frac{\phi_{i+1} - \phi_{i-1}}{2h} + O(h^2); \qquad \frac{\mathrm{d}^2\phi}{\mathrm{d}x^2} = \frac{\phi_{i+1} - 2\phi_i + \phi_{i-1}}{h^2} + O(h^2). \tag{1.4}$$

For the discretized operator L_h^c this means

$$(L_h^c \phi_h)_i \equiv \left(\frac{u}{2h} - \frac{k}{h^2}\right)\phi_{i+1} + \frac{2k}{h^2}\,\phi_i + \left(-\frac{u}{2h} - \frac{k}{h^2}\right)\phi_{i-1} = 0. \tag{1.5}$$

Let us first demonstrate the centrally discretized solution for a situation where $Pe = 500$. The grid in this example consists of 10 grid points. One observes that the discrete solution shows enthusiastic wiggles. To understand this we need the notion of a monotone operator, as discussed in Appendix A.1.



Remark In this example, the wiggles are triggered by the Dirichlet condition at $x = 1$ with a value that does not 'match' the value of ϕ in the interior. Here, we do this on purpose to show their occurrence, their origin and some strategies to prevent them. In Section 2.7 we will show how these wiggles can be suppressed by choosing a more appropriate boundary condition. However, it is good to realize that wiggles can also be induced by other reasons, e.g. near shock waves.

If the requirements for being a monotone operator are compared with the properties of the discrete scheme given by (1.5), the following can be concluded:

Theorem 1.2.1 The operator L_h^c is positive if and only if the mesh-Péclet number P satisfies

$$P \equiv \frac{|u|h}{k} \leq 2. \tag{1.6}$$

It is monotone, i.e. a maximum principle holds, if $P < 2$.

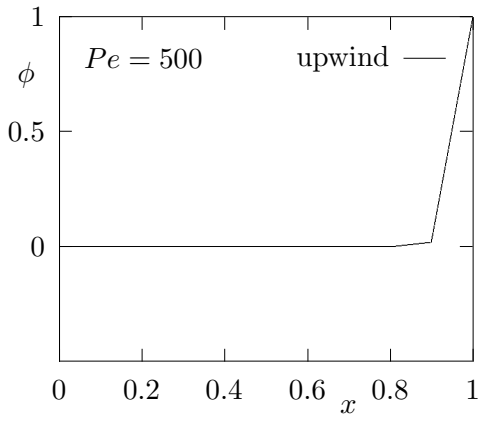
Proof For a definition of a positive operator we first refer to Appendix A.1. Further it is assumed that $u > 0$. In that case the coefficient of ϕ_{i-1}, given in (1.5), clearly is always negative, and only the coefficient of ϕ_{i+1} might give troubles. The latter coefficient is given by $u/2h - k/h^2$, which is non-positive precisely under the condition (1.6). For a maximum principle to hold, the coefficients have to be strictly negative (Th. A.1.2). □

The above condition stresses that when using central discretization the mesh width h has to be sufficiently small in order to obtain a positive operator. In the latter case the solution will be wiggle-free. A physicists reaction would be: "If you want to compute the boundary layer, then the least thing to do is to fit your grid size to the boundary-layer thickness, i.e. choose $h = k/u$." This insight is helpful, as for this choice of the grid size the mesh Péclet number becomes equal to 1 (which is smaller than 2!).

A small grid size leads to a large number of grid points and hence an expensive calculation. And maybe you are not really interested in the details within the boundary layer, i.e. convection is considered to be much more important than diffusion. In that case our physicist friend has an advice concerning the discretization of the convective term: "If the wind is blowing from a particular direction, then you only have to look in that direction to see what is coming towards you." With this physical insight, in fact he suggests to consider an alternative, one-sided discretization of the first order derivative

$$\frac{\mathrm{d}\phi}{\mathrm{d}x} = \begin{cases} \dfrac{\phi_i - \phi_{i-1}}{h} + O(h) & (u \geq 0), \\ \dfrac{\phi_{i+1} - \phi_i}{h} + O(h) & (u < 0). \end{cases} \tag{1.7}$$

(1.7) is called an 'upwind' discretization, because it is taken against the direction of the flow velocity u. This discretization is first-order accurate – in contrast with the second-order accuracy of (1.4) – but it always leads to a (monotone) positive operator irrespective of the mesh width h, as can be easily deduced (see Exercise 1.2.1). The discrete results in the same case as above indeed do not possess wiggles! But are they accurate? We will come back to this question later.



Exercise 1.2.1 Prove that, for $k > 0$, upwind discretization always leads to a positive and monotone operator.

1.2.3 Solution with central discretization

The central discretization (1.5) can be rewritten as

$$\frac{P}{2}(\phi_{i+1} - \phi_{i-1}) - (\phi_{i+1} - 2\phi_i + \phi_{i-1}) = 0 \quad (i = 1, \ldots, I-1), \tag{1.8}$$

in which the mesh-Péclet number $P = uh/k$ dominantly features. In fact, when P becomes large and ϕ remains bounded, this equation simplifies to

$$\phi_{i+1} - \phi_{i-1} \approx 0. \tag{1.9}$$

In other words, grid points two meshes apart are closely connected to each other, but they are independent of their direct neighbours. This phenomenon is called *odd-even decoupling*.

The difference equation (1.8) can be solved analytically by means of the next lemma.

Lemma 1.2.2 When $a \neq b$, the solution of the difference equation $a\,\phi_{i+1} - (a+b)\,\phi_i + b\,\phi_{i-1} = 0 \;\; (i = 1, \ldots, I-1)$, with boundary conditions $\phi_0 = T_0$ and $\phi_I = T_L$, is given by

$$\phi_i = T_0 + (T_L - T_0)\frac{1 - r^i}{1 - r^I} \text{ where } r = \frac{b}{a}. \tag{1.10}$$

Note that the solution will oscillate when $r < 0$, i.e. when a and b have a different sign. Compare this with the wiggles that can appear for non-positive operators (see Appendix A.1).

Proof Find fundamental solutions of the form r^i. These satisfy the characteristic equation $ar^2 - (a+b)r + b = 0$, which leads to $r_1 = 1$ and $r_2 = b/a$. Now, the general solution is given by $\phi_i = c_1 r_1^i + c_2 r_2^i$. The boundary conditions at $i = 0$ and $i = I$ fix the coefficients c_1 and c_2, after which (1.10) follows. □

Application of the above lemma gives the exact solution of (1.8)

$$\phi_i = T_0 + (T_L - T_0)\frac{1 - \left(\frac{1+P/2}{1-P/2}\right)^i}{1 - \left(\frac{1+P/2}{1-P/2}\right)^I}, \quad (i = 0, \ldots, I).$$

We will consider this solution for large values of P. Introduce $\epsilon \equiv 2/P \ll 1$ and apply series expansion in ϵ

$$\phi_i = T_0 + (T_L - T_0)\frac{1 - (-1)^i(1 + 2i\epsilon)}{1 - (-1)^I(1 + 2I\epsilon)} + O(\epsilon^2). \tag{1.11}$$

Distinguish the cases I is even/odd and i is even/odd.

- *I odd:*
  When I is odd (1.11) becomes

  $$\phi_i \approx T_0 + (T_L - T_0)\frac{1 - (-1)^i - 2(-1)^i i\epsilon}{2(1 + I\epsilon)},$$

  hence

  $$i \text{ even: } \phi_i \approx T_0 - (T_L - T_0)\, i\epsilon; \qquad i \text{ odd: } \phi_i \approx T_L - (T_L - T_0)(I - i)\epsilon.$$

The solution in even grid points fits to the left-hand-side boundary condition, whereas the odd points fit to the right-hand-side condition. This is a concrete example of the odd/even decoupling announced above.

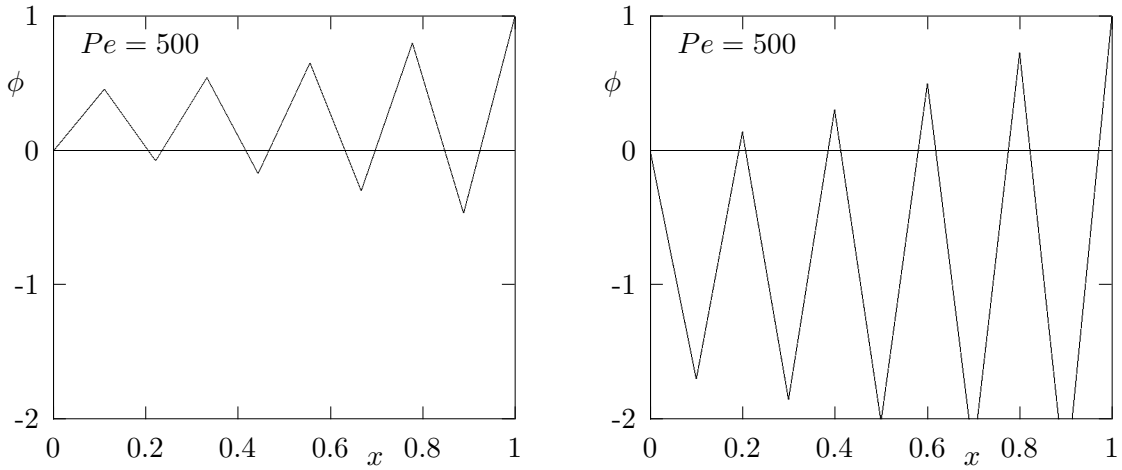
- *I even:*
  When I is even we have

$$\phi_i \approx T_0 + (T_L - T_0)\frac{1-(-1)^i(1+2i\epsilon)}{-2I\epsilon},$$

hence

$$i \text{ even: } \phi_i \approx T_0 + (T_L - T_0)\frac{i}{I}; \quad i \text{ odd: } \phi_i \approx T_0 + (T_L - T_0)\left(\frac{-1}{I\epsilon} - \frac{i}{I}\right).$$

The solution in the even grid points fits to both boundary conditions. Odd grid points look awful: the solution there approaches infinity when $\epsilon \to 0$. Note that ϕ_i is no longer bounded, hence (1.9) does not describe the behaviour.



Central discretization for $Pe = 500$: I odd (left) and I even (right).

1.2.4 Solution with upwind discretization

When an upwind discretization (1.7) is applied, the discrete equation becomes (assuming $u > 0$)

$$P(\phi_i - \phi_{i-1}) - (\phi_{i+1} - 2\phi_i + \phi_{i-1}) = 0, \ \phi_0 = T_0, \ \phi_I = T_L.$$

The solution again follows from (1.10)

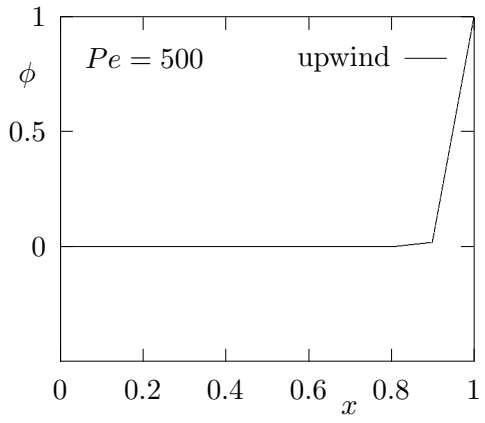
$$\phi_i = T_0 + (T_L - T_0)\frac{1-(1+P)^i}{1-(1+P)^I}.$$

For P large we have

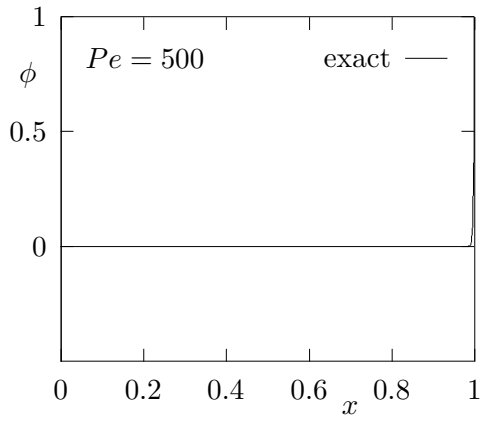
$$\phi_i \approx T_0 + (T_L - T_0)P^{i-I},$$

which for $i < I$ is approximately equal to T_0, whereas $\phi_I = T_L$. The solution does not possess wiggles and looks more friendly than the central solution. But the boundary layer is too thick! The two figures below show the upwind solution and the exact solution for a situation

with $I = 10,\ k = 0.002,\ T_0 = 0$ and $T_L = 1$.



Upwind discretization



Exact solution

1.3 Artificial diffusion

1.3.1 Second-order diffusion

If we compare the (second-order) central discretization with the (first-order) upwind discretization through

$$u\,\frac{\phi_i - \phi_{i-1}}{h} \equiv u\,\frac{\phi_{i+1} - \phi_{i-1}}{2h} - \frac{uh}{2}\,\frac{\phi_{i+1} - 2\phi_i + \phi_{i-1}}{h^2},$$

it is observed that upwind discretization of (1.2) yields the same discrete equation as central discretization of

$$u\,\frac{\mathrm{d}\phi}{\mathrm{d}x} - \left(k + \frac{uh}{2}\right)\frac{\mathrm{d}^2\phi}{\mathrm{d}x^2} = 0. \tag{1.12}$$

When comparing the centrally discretized solution of (1.2) with the upwind discretized solution of (1.2), the formulation (1.12) can be used as an intermediate step.

upwind discretization of (1.2)	$\equiv$	central discretization of (1.12)	$\longleftrightarrow$	central discretization of (1.2)

It is observed that (1.12) in comparison with (1.2) contains an additional term in which the diffusion coefficient is increased with $k_a = uh/2$. This increase is called *artificial diffusion*. When the artificial diffusion dominates the real diffusion, and when one is interested in the effects of diffusion, then the usefulness of the upwind discretized solution decreases. But be aware! We have silently assumed that the other discretization errors are small and that they do not play a role in the above. If this is not the case then the judgement on upwind discretization can be more positive. Also when diffusion is not relevant for the physics there is little harm in increasing the diffusion, making upwind discretization an acceptable choice.

The advantage of upwind discretization is that a (monotone) positive operator is created, which is easily treated iteratively (see Appendix A.2). Also other choices for the artificial viscosity k_a lead to a positive operator. To investigate this, let us discretize the equation

$$u\,\frac{\mathrm{d}\phi}{\mathrm{d}x} - (k+k_a)\,\frac{\mathrm{d}^2\phi}{\mathrm{d}x^2} = 0 \tag{1.13}$$

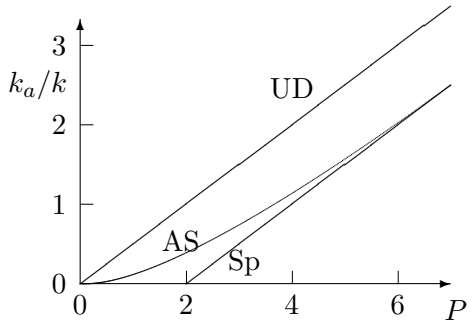
for arbitrary k_a using central discretization. We obtain

$$(L_h^a\phi_h)_i \equiv \frac{uh}{2}(\phi_{i+1}-\phi_{i-1}) - (k+k_a)\,(\phi_{i+1}-2\phi_i+\phi_{i-1}) = 0.$$

With reference to Theorem 1.2.1, the operator L_h^a is a positive operator if and only if

$$k+k_a \geq \frac{|u|h}{2} \quad\Leftrightarrow\quad k_a \geq \frac{|u|h}{2} - k.$$

The upwind discretization with $k_a = |u|h/2$ satisfies this criterion. In the literature also other choices can be found. The figure below shows the quantity k_a/k, for a number of methods, as a function of $P = |u|h/k$.



UD = upwind discretization

Sp = Spalding (1972):
$k_a = \max\,(0, \frac{|u|h}{2} - k)$

AS = Allen & Southwell (1955)

A special choice for k_a has been described for the first time by Allen & Southwell (1955), and reinvented many times. Gresho & Lee (1981) call it the 'smart upwind' method. Other names are the 'locally exact' method, and the 'exponential scheme'. This special choice of k_a can be found by comparing (1.3) and (1.10). One notes that for $r = e^{Pe/I} = e^P$ the discrete solution (1.10) and the analytic solution (1.3) in the grid points are equal. This situation arises when (1.13) is centrally discretized with an artificial diffusion coefficient (assume $u > 0$)

$$k_a = \frac{uh}{2}\,\coth\,\frac{P}{2} - k = k\left(\frac{P}{2}\,\coth\,\frac{P}{2} - 1\right). \tag{1.14}$$

Proof After central discretization of (1.13), an equation as in Lemma 1.2.2 is obtained with

$$a \equiv \frac{u}{2h} - \frac{k+k_a}{h^2} \quad\text{and}\quad b \equiv -\frac{u}{2h} - \frac{k+k_a}{h^2}.$$

Requiring that $z \equiv b/a = e^P$, we can work backward to 'reconstruct' k_a:

$$-\frac{u}{2h} - \frac{k+k_a}{h^2} = e^P\left(\frac{u}{2h} - \frac{k+k_a}{h^2}\right) \quad\Rightarrow\quad k+k_a = \frac{uh}{2}\left(\frac{e^P+1}{e^P-1}\right) = \frac{uh}{2}\left(\frac{e^{P/2}+e^{-P/2}}{e^{P/2}-e^{-P/2}}\right).$$

□

For small P the expression between parentheses in (1.14) approaches zero, hence in this case no artificial diffusion is added. The discrete solution will now approach the solution of the centrally discretized problem. For large P, k_a behaves like $k_a \approx \frac{1}{2}uh - k$. The solution then resembles the upwind discretized solution. Yet, for the above one-dimensional convection-diffusion equation with constant coefficients the exact solution is obtained. In more than one dimension, nothing similar has been found yet.

Artificial diffusion is often applied in situations with $P > 2$, i.e. $\frac{1}{2}h > k/u =$ the boundary-layer thickness, to obtain a system of discrete equations which is easier tractable numerically. When the solution is smooth - the amount of diffusion is not important then - this does not matter very much. However, when the solution should possess a boundary-layer character the mesh width simply is too large to resolve details in the boundary layer. No choice of k_a can repair this situation, not even (1.14) which gives the exact solution! A smooth solution is created, however its boundary layer is too thick[2]. This can be seen only by comparing with the exact solution. The left-hand-side figure on page 7 looks nice, but comparison with the exact solution shows that the boundary layer is not computed correctly. In such a situation the centrally discretized solution gives a warning by showing violent wiggles (see the figures on page 6); compare Gresho & Lee (1981). When the details in the boundary layer are relevant, the only remedy is to refine the mesh width such that at least a few grid points are located inside the boundary layer. Then automatically $P \leq 2$ inside the boundary layer (check this!). Outside the boundary layer $P > 2$, but that need not be critical (see Section 1.5).

1.3.2 Higher-order additives

The wiggle-dependence of central discretization of a first-order derivative also can be diminished by adding a term of the form $h^2\,\mathrm{d}^3\phi/\mathrm{d}x^3$ (for $u > 0$)[3]

$$\frac{\mathrm{d}\phi}{\mathrm{d}x} = \frac{\phi_{i+1} - \phi_{i-1}}{2h} - \lambda\,\frac{\phi_{i+1} - 3\phi_i + 3\phi_{i-1} - \phi_{i-2}}{h} + \tau_h, \tag{1.15}$$

with

$$\tau_h = h^2\left(\lambda - \frac{1}{6}\right)\phi_{xxx} + O(h^3).$$

Here we have a family of upwind-biased λ-schemes[4], which all are second-order accurate; for $\lambda = 1/6$ we even have a third-order discretization. Special cases are further $\lambda = 0$ (central discretization), $\lambda = 1/2$ (second-order upwind: the B3-scheme in which ϕ_{i+1} drops out) and $\lambda = 1/8$ (the QUICK method).

Why $\lambda = 1/8$ is a special value, can be seen as follows. Consider in the figure below the points R and L which lie halfway $[i, i+1]$, and $[i-1, i]$, respectively. We approximate the

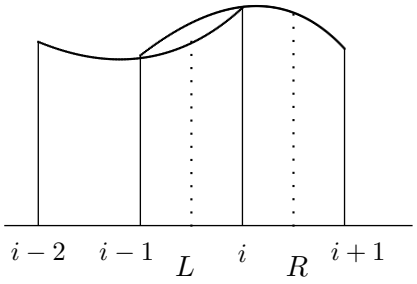
[2] With upwind diffusion $k_a \equiv uh/2$, the thickness of the (artificial) boundary layer becomes $k_a/u = h/2$, i.e. it can be resolved by the grid.

[3] An upwind-biased second-order discretization of a third-order derivative is given by

$$\frac{\mathrm{d}^3\phi}{\mathrm{d}x^3} = \frac{\phi_{i+1} - 3\phi_i + 3\phi_{i-1} - \phi_{i-2}}{h^3}.$$

[4] Above upwind-biased schemes are also used, in disguised form, in compressible flow, where they are called κ-schemes; see e.g. Hirsch (1990) or Wesseling (2001). They are equivalent when $\kappa = 1 - 4\lambda$.

derivative in i in a finite-volume fashion through



$$\frac{\mathrm{d}\phi}{\mathrm{d}x} = \frac{\phi_R - \phi_L}{h}.$$

The values ϕ_R and ϕ_L are determined via quadratic interpolation on the intervals $[i-1, i+1]$, and $[i-2, i]$ respectively (assuming $u > 0$). In this way

$$\phi_R = \frac{\phi_{i+1} + \phi_i}{2} - \frac{\phi_{i+1} - 2\phi_i + \phi_{i-1}}{8},$$

with a similar formula for ϕ_L. QUICK stands for Quadratic Upstream Interpolation for Convective Kinematics (Leonard 1979a,b).

After having added a second-order derivative and a third-order derivative to 'smooth' central discretization, it sounds logical that a fourth-order derivative (multiplied by h^3) also might be useful. Indeed this is the case. Jameson *et al.* (1981) have introduced this approach for solving the inviscid Euler equations, but we will not go into further detail here. Another way of getting rid of the wiggles is to apply monotone discretization methods, where keywords like 'limiters' and 'TVD (total variation diminishing)' are issues; see e.g. Chapters 20 and 21 of Hirsch (1990) or Chapter 9 of Wesseling (2001).

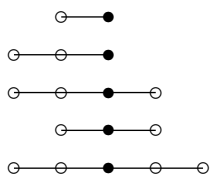
Exercise 1.3.1 Prove that $\lambda \geq \max(\frac{1}{2} - \frac{k}{|u|h}, 0)$ is a sufficient condition for the upwind-biased methods from (1.15) to be wiggle-free. Show that the QUICK method is wiggle-free for $P \leq 8/3$. Hint: Try fundamental solutions of the form r^i, and monitor the sign of r.

1.4 Preliminary trade-off

The trade-off between the methods presented above (and the methods still to be presented) is a delicate matter. An important role is played by the ratio between the mesh size and the length scales of the solution.

To give a first impression we have compared a number of methods:

i) the first-order upwind method (1.7);
ii) the second-order upwind method (1.15) with $\lambda = 1/2$;
iii) the QUICK method (1.15) with $\lambda = 1/8$;
iv) the second-order central method (1.4);
v) the fourth-order central method (see Section 1.7).

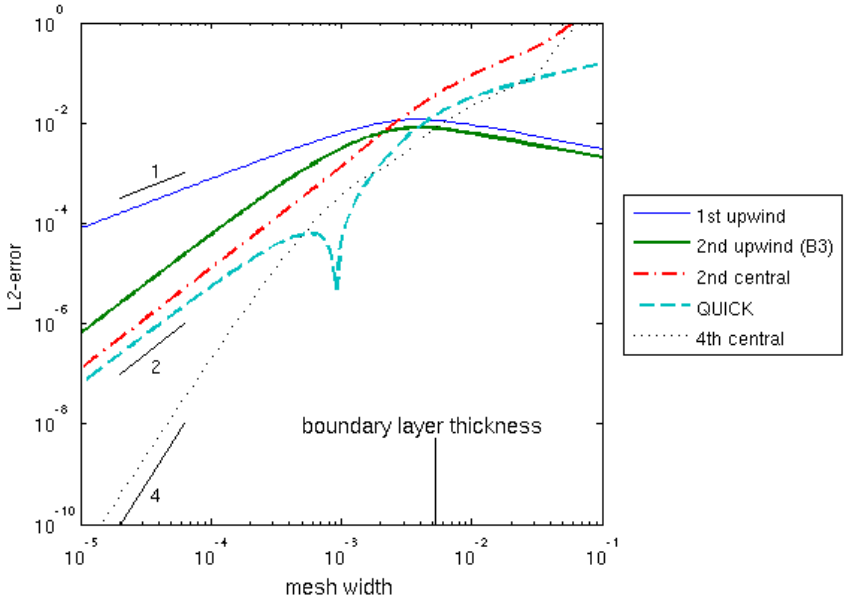


The difference stencils of methods ii), iii) and v) are too large near the boundaries. Therefore, where necessary in the first and last grid points a central discretization has been applied. Except for the fourth-order method, the diffusive term has been discretized with the second-order discretization as used in (1.4). Thus, it makes not much sense to include the third-order λ-scheme with $\lambda = 1/6$ in this comparison, as the second-order error from diffusion will dominate for small grid sizes; anyway its results do not differ dramatically from those for $\lambda = 1/8$.

The convection-diffusion equation (2.3) has been solved with $k = 10^{-3}$. Its boundary layer has thickness δ_{995} (defined as the location where the solution reaches 99.5% of the outer solution) of about 0.0053. The computations have been performed on uniform grids for a large range of mesh sizes, varying from $h = 10^{-1}$ (with which the boundary layer cannot be resolved at all) to $h = 10^{-5}$ (with over 500 points inside the boundary layer). In the figure the difference (as a discrete L_2-norm) between the analytic solution and the discrete solution

$$\|y_{\text{exact}} - y_h\|_h = \left\{ \frac{1}{N} \sum_{i=0}^{N} (y_{\text{exact}}(x_i) - y_h(x_i))^2 \right\}^{1/2}$$

has been plotted. This quantity is a measure for the error of the discrete solution in the grid points. Where the grid is too coarse to resolve the analytic solution, this quantity gives a flattered image: what happens (or should happen) between the grid points remains invisible.



Accuracy for given grid size We start with some conclusions concerning the discretization error of the methods investigated.

- There is no method that performs well for all mesh sizes.
- For coarse grids, only the first- and second-order upwind methods are useful. The error of the first-order upwind method is a bit flattered because outside the boundary layer the solution is constant. In a less trivial situation it performs worse.
- When the grid is refined until the boundary layer can be resolved, the other three methods become more attractive. Both central methods behave very regular. The QUICK method is irregular, with a favourable discretization error just in the accuracy range around 10^{-3}–10^{-4} which is relevant for practical calculations. Whether this is a structural property of QUICK is unknown[5].

[5] In October 1996, I have discussed this matter with Brian Leonard, the 'inventor' of QUICK (Leonard 1979a), but he does not know either whether this behaviour is systematic. He also told me the story behind the intriguing title of the conference paper Leonard (1979b).

- For very fine grids the fourth-order method is the most accurate.

Computational effort Next, the effort to solve the discrete system of equations should be considered. It will be clear that a larger difference stencil will lead to a less sparse matrix, making direct solution more expensive. Yet, for small (2D) problems direct methods will be the fastest. When the system is solved iteratively, the upwind method is very easy to handle. The other methods require a more sophisticated iterative treatment, which usually will be more expensive. However, when this iteration process can be combined with other iterative processes, e.g. treating non-linearity, the computational pain will be relieved. Over the years, many useful methods have been presented in the literature (for the developments at RUG see e.g. Wubs and Thies 2011).

Maintainability A final aspect to be taken into account is the simplicity of the method and herewith the simplicity of the computer code. Methods with a larger stencil require special measures near the boundaries, introducing many exceptional cases, and increasing the chance for errors: the code becomes less readable and less maintainable.

Trade-off

- For engineering accuracy (say 10^{-3}) I would prefer a second-order central discretization: a small stencil, and easy to implement. Of course, the grid size has to be adapted to the behaviour of the solution: too coarse meshes can lead to large oscillations, as explained above. The QUICK method, with its larger stencil, is still a mystery to me.

- For coarse-grid calculations only the upwind methods qualify. First-order upwind is easily handled, but its results are accordingly. Second-order upwind is remarkably better, but it has the disadvantage of a larger stencil. It is the only method with reasonable performance for a large range of mesh sizes.

- Only when one is interested in four or more figures accuracy the fourth-order methods are of interest. One should take into account that the equations to be solved often will be an approximation of reality containing modelling errors. There is no sense in reducing the discretization error too far below this modelling error.

In the next sections we will discuss more advanced discretization methods with which their price/performance ratio can be improved.

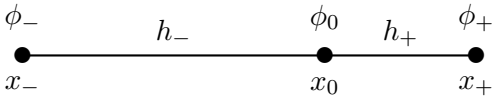
1.5 Non-uniform grids

Thus far grids have been used in which, per coordinate direction, the mesh size is constant. In practice often grids with varying mesh size are used. Their purpose is to describe the solution, at prescribed accuracy, with as little grid points as possible. In this way the discrete systems to be solved become smaller, and in general – but not always (cf. Exercise 1.5.1)! – can be solved cheaper.

1.5.1 Discretization

We start with a study of the local discretization error on a non-equidistant grid. Hereto we use a one-dimensional example in which a first-order and a second-order derivative are discretized.

Consider the following triple of grid points, with function values ϕ_-, ϕ_0 and ϕ_+, and at mutual distances h_- and h_+



In the middle grid point the derivatives ϕ_x and ϕ_{xx} are approximated with finite-difference formulas in which only ϕ_-, ϕ_0 and ϕ_+ appear. We start with the Taylor series for ϕ_+ and ϕ_-:

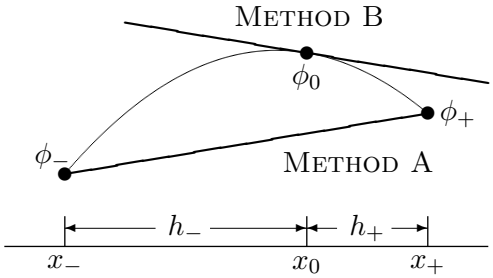
$$\phi_+ = \phi_0 + h_+\phi_x + \tfrac{1}{2}h_+^2\phi_{xx} + \tfrac{1}{6}h_+^3\phi_{xxx} + \cdots , \tag{1.16}$$

$$\phi_- = \phi_0 - h_-\phi_x + \tfrac{1}{2}h_-^2\phi_{xx} - \tfrac{1}{6}h_-^3\phi_{xxx} + \cdots . \tag{1.17}$$

When we simply subtract (1.16) and (1.17) we obtain (Method A)

$$\begin{aligned} \phi_x &= \frac{\phi_+ - \phi_-}{h_+ + h_-} - \tfrac{1}{2}(h_+ - h_-)\,\phi_{xx} - \tfrac{1}{6}\frac{h_+^3 + h_-^3}{h_+ + h_-}\,\phi_{xxx} + \cdots \\ &= \frac{h_+}{h_+ + h_-}\frac{\phi_+ - \phi_0}{h_+} + \frac{h_-}{h_+ + h_-}\frac{\phi_0 - \phi_-}{h_-} + \cdots . \end{aligned} \tag{1.18}$$

On uniform grids the term $(h_+ - h_-)\phi_{xx}$ vanishes; this term is related to the stretching of the grid (see below). Geometrically this estimate for the derivative in the i-th grid point is generated by means of a linear interpolation of ϕ between the adjacent grid points (for a finite-volume interpretation see Section 1.6).



By combining (1.16) and (1.17) in a different way, the term with ϕ_{xx} can be eliminated beforehand. Take $h_-^2 \times (1.16) - h_+^2 \times (1.17)$, then one obtains (Method B)

$$\begin{aligned} \phi_x &= \frac{h_-^2\,\phi_+ + \left(h_+^2 - h_-^2\right)\phi_0 - h_+^2\,\phi_-}{h_+h_-(h_+ + h_-)} - \tfrac{1}{6}h_+h_-\phi_{xxx} + \cdots \\ &= \frac{h_-}{h_+ + h_-}\frac{\phi_+ - \phi_0}{h_+} + \frac{h_+}{h_+ + h_-}\frac{\phi_0 - \phi_-}{h_-} + \cdots . \end{aligned} \tag{1.19}$$

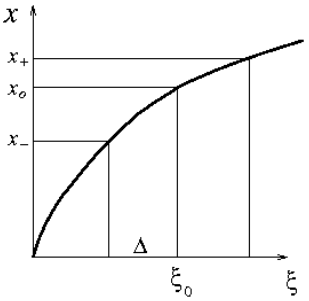
First observe that the weights of the forward and backward derivatives have been interchanged in comparison with (1.18). Further, the local discretization error in (1.19) looks better than the one in (1.18). Not only a term with ϕ_{xx} is missing, but also the coefficient of ϕ_{xxx} is

less or equal the corresponding coefficient in (1.18). Again a geometric interpretation can be given. Draw a parabola through the three points ϕ_-, ϕ_0 and ϕ_+ (Lagrange interpolation), then the tangent to this parabola in ϕ_0 gives the estimate (1.19). Note that, because of these Taylor-series considerations, (1.19) is the 'standard' central discretization for a first-order derivative. However, in the next section we will see that it does possess serious drawbacks.

For the approximation of ϕ_{xx}, only one feasible option is available: combine $h_- \times (1.16) + h_+ \times (1.17)$

$$\phi_{xx} = \frac{h_-\,\phi_+ - (h_+ + h_-)\,\phi_0 + h_+\,\phi_-}{\frac{1}{2}h_+h_-(h_+ + h_-)} - \tfrac{1}{3}(h_+ - h_-)\,\phi_{xxx} + \cdots . \tag{1.20}$$

Again the difference $h_+ - h_-$ appears. In the literature it is often called a first-order term, but it can equally well be called a second-order term: this depends on the way in which the mesh width approaches zero. When h_+ and h_- approach zero with $h_+/h_- = \text{constant} \neq 1$, it is a first-order term. It is a second-order term when the grid is obtained through a transformation, $x = f(\xi)$, in which the ξ-interval is divided uniform with mesh width Δ. From a Taylor series around ξ_0 (corresponding with the central point) we get



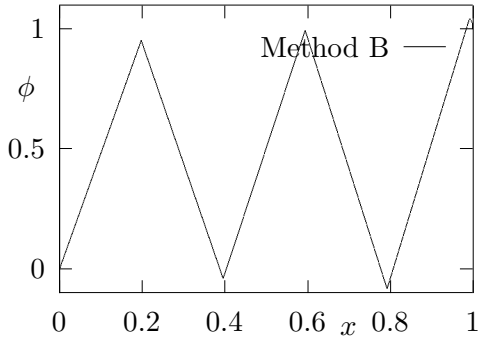
$$\begin{aligned} h_+ &\equiv x_+ - x_0 = \Delta f'(\xi_0) + \tfrac{1}{2}\Delta^2 f''(\xi_0) + \cdots , \\ h_- &\equiv x_0 - x_- = \Delta f'(\xi_0) - \tfrac{1}{2}\Delta^2 f''(\xi_0) + \cdots , \end{aligned}$$

hence

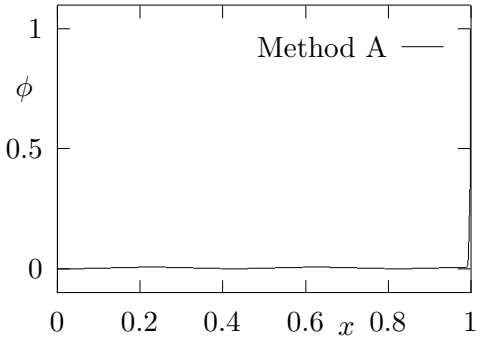
$$h_+ - h_- = \Delta^2 f''(\xi_0) + \cdots \quad \text{and} \quad \frac{h_+}{h_-} = 1 + \Delta\frac{f''}{f'}(\xi_0) + \cdots .$$

It will be clear that $f''(\xi)$ controls the amount of stretching of the grid.

1.5.2 Numerical benchmark



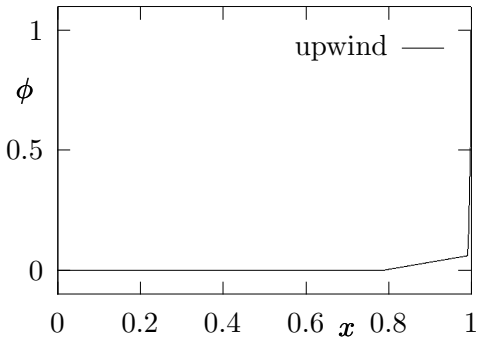
Discretization according to Method B (1.19) yields the above result.



Discretization by Method A (1.18), which looks less promising then (1.19) when measuring the local truncation error, produces much better results.

The difference between both discretization methods A and B will be demonstrated with a few examples. Consider the model problem (1.2) on the interval $0 \leq x \leq 1$, with $u = 1$ and

For comparison we show the discrete solution from upwind discretization.



$k = 0.002$ and boundary conditions $\phi(0) = 0$ and $\phi(1) = 1$. The exact solution is shown on page 7. In the examples we use a grid given by

$$x_i = \{0,\, 0.2-k,\, 0.4-2k,\, 0.6-3k,\, 0.8-4k,\, 1-5k,\, 1-4k,\, 1-3k,\, 1-2k,\, 1-k,\, 1\}, \quad (1.21)$$

hence for $k = 0.002$ we have five grid cells of size about 0.2 and five grid cells of size 0.002.

1.5.3 Discussion: steady

The above example gives an impression of the systematic behaviour of both methods; more examples can be found in Veldman *et al.* (1991, 1993) and Verstappen and Veldman (1997, 1998, 2003). We will now give a fundamental explanation of the observed behaviour.

To get a feeling of the relation between the *local* discretization (= truncation) error (as in (1.18) and (1.19)) and the *global* discretization error we have to consider (see Appendix A.1)

$$\boldsymbol{\phi}_{exact} - \boldsymbol{\phi}_h = \boldsymbol{L}_h^{-1}\boldsymbol{\tau}_h. \quad (1.22)$$

This relation shows that the global error is determined by multiplying the local error with the inverse of the coefficient matrix. When the diagonal of $\boldsymbol{L}_h$ is weakened, roughly spoken, some eigenvalues may move towards zero herewith 'increasing' $\boldsymbol{L}_h^{-1}$. In the examples shown, for Method B this increase of $\boldsymbol{L}_h^{-1}$ apparently overrules the decrease of $\boldsymbol{\tau}_h$. Let us analyse this in more detail.

Consider first the coefficient matrix $\boldsymbol{L}_A$ of Method A, defined from (1.18) and (1.20), which will be scaled by a diagonal matrix $\boldsymbol{H}$ incorporating the local grid size

$$\boldsymbol{M}_A = \boldsymbol{H}\boldsymbol{L}_A \quad \text{where} \quad \boldsymbol{H} = \operatorname{diag}\left\{\frac{h_- + h_+}{2}\right\}. \quad (1.23)$$

A simple calculation reveals that

$$2\boldsymbol{M}_A = \begin{pmatrix} \ddots & \ddots & \ddots & & \\ -u-\frac{2k}{h_{--}} & \frac{2k(h_{--}+h_-)}{h_{--}h_-} & u-\frac{2k}{h_-} & & \\ & -u-\frac{2k}{h_-} & \frac{2k(h_-+h_+)}{h_-h_+} & u-\frac{2k}{h_+} & \\ & & -u-\frac{2k}{h_+} & \frac{2k(h_++h_{++})}{h_+h_{++}} & u-\frac{2k}{h_{++}} \\ & & \ddots & \ddots & \ddots \end{pmatrix}, \quad (1.24)$$

where h_{--}, h_-, h_+ and h_{++} are adjacent mesh sizes.

In the matrix $\boldsymbol{M}_A$ the diffusive term gives a purely symmetric contribution and the convective term gives a purely skew-symmetric contribution, in line with the analytical property of diffusion and convection[6], respectively. Therefore a method with such a property is called *symmetry preserving*. This property allows to prove that the coefficient matrix $\boldsymbol{M}_A$ can never become singular, no matter how the grid is chosen. We formulate this in the next theorem.

Theorem 1.5.1 For Method A, the eigenvalues of the scaled coefficient matrix $\boldsymbol{M}_A$ lie in the right half plane (i.e. the matrix is positive-stable); hence this matrix is nonsingular.

Proof Bendixson's theorem A.2.5 states that all eigenvalues of a matrix lie in the rectangle formed by the eigenvalues of the symmetric part and the eigenvalues of the skew-symmetric part. As just remarked, the symmetric part of $\boldsymbol{M}_A$ is generated by the diffusive term. This part is weakly diagonally dominant, and hence positive definite. Thus the rectangle just mentioned lies fully in the right half plane, and all eigenvalues of $\boldsymbol{M}_A$ have a positive real part. In particular $\boldsymbol{M}_A$ is nonsingular. □

Also several related matrices are nonsingular, as formulated in the following theorem:

Theorem 1.5.2 For Method A, also the eigenvalues of the non-scaled coefficient matrix $\boldsymbol{L}_A$ and of the shifted Jacobi matrix $(\operatorname{diag}\boldsymbol{L}_A)^{-1}\boldsymbol{L}_A = (\operatorname{diag}\boldsymbol{M}_A)^{-1}\boldsymbol{M}_A$ are lying in the positive half plane. Thus these matrices are positive-stable and hence nonsingular.

Proof First observe that, trivially, the diagonal matrix $\boldsymbol{H}$ is positive definite. Further, the symmetric part of $\boldsymbol{M}_A$ is positive definite. Then Lemma A.2.8 makes the non-scaled coefficient matrix $\boldsymbol{L}_A = \boldsymbol{H}^{-1}\boldsymbol{M}_A$ positive-stable. Also, $\boldsymbol{D}_A \equiv \operatorname{diag}\boldsymbol{M}_A$ is positive definite. This, similarly, makes $\boldsymbol{D}_A^{-1}\boldsymbol{M}_A$ positive-stable. □

In particular, the coefficient matrix $\boldsymbol{L}_A$ of Method A is never singular and its global error (1.22) can never become infinite, irrespective of the chosen computational grid. In the RUG lecture notes "Computational Methods of Science" it is explained how this is related to the stability of the discrete operator $\boldsymbol{L}_A$.

Method B does not possess such pleasant properties. On the contrary! The diagonal entries of this method (before scaling) are given by

$$\frac{u\,(h_+ - h_-) + 2k}{h_- h_+}, \tag{1.25}$$

from which we see that the convective term contributes to the diagonal. The diagonal can easily become negative; note that in our example $h_+ < h_-$, which is a natural situation for $u > 0$ in view of the outflow boundary layer that may appear. Hereafter $\boldsymbol{D}_B^{-1}\boldsymbol{M}_B$ is found to be no longer positive-stable (see figure). When the lowering of the diagonal is stronger, some eigenvalues may move to the left half plane and the (scaled) coefficient matrices $\boldsymbol{L}_B$ and $\boldsymbol{M}_B = \boldsymbol{H}\boldsymbol{L}_B$ can even become singular (and the global error (1.22) becomes infinite)!

[6] In one dimension, and omitting boundary effects, the skew-symmetry of convection basically follows from the integration-by-parts formula: $\int u'v\,\mathrm{d}x = -\int uv'\,\mathrm{d}x$, whereas the symmetry of diffusion follows from $\int u''v\,\mathrm{d}x = -\int u'v'\,\mathrm{d}x = \int uv''\,\mathrm{d}x$.

To illustrate this we show one of the eigenvalues of $\boldsymbol{M}_B$ and of $\boldsymbol{D}_B^{-1}\boldsymbol{M}_B$ as a function of k, using the grid (1.21). Already for $k = 1/20$, where the stretching 'h_+/h_-' in the irregular grid point equals 1/3, the shifted Jacobi matrix is singular (a diagonal element becomes zero). For $k = k_M \approx 0.0084$, where the stretching is approximately 0.04, the coefficient matrix itself becomes singular.

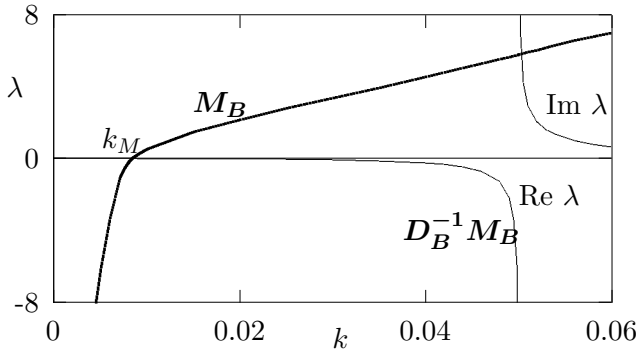


Figure 1.1: *One of the eigenvalues of the shifted Jacobi matrix* $\boldsymbol{D}_B^{-1}\boldsymbol{M}_B$ *becomes infinite for* $k = 0.05$. *The matrix* $\boldsymbol{M}_B$ *itself becomes singular for* $k \approx 0.0084$.

Remark Manteuffel and White (1986) have proven theoretically that on grids where the ratio between the smallest and the largest mesh size is bounded during grid refinement, both Method A and Method B are second-order accurate. In view of the above observations, this again is an acknowledgement of the power of Method A, where the local discretization error at first sight would suggest only first-order accuracy.

Remark In the above approach, the analytic skew-symmetry of the convective operator has been preserved in its discrete counterpart. This is an example of a general strategy to preserve (i.e. to mimic) properties of the analytic equations. In the literature, this general approach is known as *mimetic* discretization; other names are compatible or symmetry- (or structure-) preserving discretization.

1.5.4 Discussion: unsteady

In an unsteady setting

$$\frac{\mathrm{d}\boldsymbol{\phi}_h}{\mathrm{d}t} + \boldsymbol{L}_h\boldsymbol{\phi_h} = 0 \tag{1.26}$$

it is possible to talk about conservation of certain solution properties with time. Let us define the 'energy' $||\boldsymbol{\phi}_h||_h^2$ of the solution in such a way that it incorporates the local grid size: in matrix-vector notation

$$||\boldsymbol{\phi}_h||_h^2 \equiv \boldsymbol{\phi}_h^* \boldsymbol{H} \boldsymbol{\phi}_h. \tag{1.27}$$

When $\boldsymbol{H}$ is positive definite, this is a genuine vector norm.

Theorem 1.5.3 The solution of (1.26) conserves energy (1.27) if and only if the scaled coefficient matrix $\boldsymbol{H}\boldsymbol{L}_h$ is skew-symmetric. Furthermore the energy is decreasing if and only if $\boldsymbol{H}\boldsymbol{L}_h$ is positive real.

Proof The evolution of the energy (1.27) in time reads (note that $\boldsymbol{H}$ is symmetric)

$$\frac{\mathrm{d}}{\mathrm{d}t}||\boldsymbol{\phi}_h||_h^2 = \frac{\mathrm{d}\boldsymbol{\phi}_h^*}{\mathrm{d}t}\boldsymbol{H}\boldsymbol{\phi}_h + \boldsymbol{\phi}_h^*\boldsymbol{H}\frac{\mathrm{d}\boldsymbol{\phi}_h}{\mathrm{d}t} = -(\boldsymbol{L}_h\boldsymbol{\phi}_h)^*\boldsymbol{H}\boldsymbol{\phi}_h - \boldsymbol{\phi}_h^*\boldsymbol{H}\boldsymbol{L}_h\boldsymbol{\phi}_h = -\boldsymbol{\phi}_h^*(\boldsymbol{H}\boldsymbol{L}_h + (\boldsymbol{H}\boldsymbol{L}_h)^*)\boldsymbol{\phi}_h.$$

The right-hand side vanishes if and only if $\boldsymbol{H}\boldsymbol{L}_h$ is skew-symmetric. It is negative for all $\boldsymbol{\phi} \neq \boldsymbol{0}$ if and only if the symmetric part of $\boldsymbol{H}\boldsymbol{L}_h$ is positive definite, i.e. $\boldsymbol{H}\boldsymbol{L}_h$ is positive real. □

The scaled coefficient matrix $\boldsymbol{M}_A(= \boldsymbol{H}\boldsymbol{L}_A)$ of Method A satisfies the latter condition in the theorem, once again explaining the favourable properties of this discretization method. On the other hand, the scaled coefficient matrix of Method B does not satisfy the conditions in the theorem. A fortiori, as soon as one of the eigenvalues of the coefficient matrix becomes negative (or has negative real part), the semi-discretized time-dependent formulation (1.26) is unstable.

Exercise 1.5.1 Write down the discretization of the unsteady convection-diffusion equation using forward-Euler in time and Method B in space. Use a grid with only one interior grid point. Show that the time integration is unstable when $2k < u(h_- - h_+)$. Note that here we have an example where reduction of the number of grid points in(!)creases the computational effort. Next apply Method A and choose the single interior grid point as $1 - 2k$. What do you think of the discrete steady solution in this case?

1.5.5 Iterative solution

Another consequence of the favourable properties proven in Theorem 1.5.2, in particular for the shifted Jacobi matrix, is that many iterative methods can be used to solve the discrete system for Method A, for example JOR and SOR (see Appendix A.2). And also on this issue Method B gets into trouble. Already when its diagonal becomes singular (see Figure 1.1), SOR and similar iterative solution methods can no longer be used to solve the equations from Method B.

As an example, consider the one-dimensional convection-diffusion equation discretized with central differences from Section 1.2:

$$(-\frac{P}{2} - 1)\ \phi_{i-1} + 2\phi_i + (\frac{P}{2} - 1)\ \phi_{i+1} = 0,$$

with Dirichlet boundary conditions (P is the mesh-Péclet number). The Jacobi matrix "$I - D^{-1}A$" (see Appendix A.2) reads

$$\mathrm{diag}\left[\frac{P}{4} + \frac{1}{2},\ 0,\ -\frac{P}{4} + \frac{1}{2}\right].$$

The eigenvalues of this Jacobi matrix are given by (see Lemma A.2.10)

$$\mu_J = \frac{1}{2}\sqrt{4 - P^2}\ \cos(\frac{i\pi}{I});\ i = 1, 2, \ldots, I - 1.$$

These eigenvalues are real if and only if $P \leq 2$, which corresponds precisely with the 'wiggle-limit'. For $P > 2$ they are purely imaginary, with an imaginary part that is proportional to P. From (A.19) we infer that for JOR the optimum relaxation factor decreases quadratically with P: $\omega_{\mathrm{JOR,opt}} \sim \frac{4}{P^2}$. The spectral radius is correspondingly close to 1, $\rho_{\mathrm{JOR,opt}} \sim 1 - \frac{2}{P^2}$,

hence convergence is not really fast.

This example shows that a central discretization of a large convective term causes divergence of the Jacobi method. Severe underrelaxation is required to obtain convergence; professional skill is required to determine a suitable relaxation factor. On the other hand, the upwind method does not pose any iterative problem whatsoever (since the matrix remains diagonally dominant). For many researchers this iterative convenience is the reason to choose upwind . . . !

The convergence of SOR is also subject to good skills in choosing its relaxation factor, although by comparing (A.19) and (A.21) we see that in this example SOR converges significantly faster than JOR. The difference is much larger than the factor two between Gauss-Seidel and Jacobi. A little arithmetic with (A.21) yields that for large mesh-Péclet numbers P we have $\omega_{\mathrm{SOR,opt}} \sim \frac{4}{P}$ with $\rho_{\mathrm{SOR,opt}} \sim 1 - \frac{4}{P}$. In the computer exercise in Appendix B.3 we will see how 'tricky' this slow convergence can be.

Many other iterative solution methods require the eigenvalues of either $\boldsymbol{M}$ or $(\mathrm{diag}\,\boldsymbol{M})^{-1}\boldsymbol{M}$ to lie in the stable right-half plane. For Method B this cannot be guaranteed, but Theorem 1.5.2 shows that Method A always satisfies this requirement.

1.6 Finite-volume discretization

In a finite-volume discretization each grid point is surrounded by a cell, or control volume, in which the weak form of the conservation law is applied. When applied to a control volume Ω_h with boundary Γ_h a general conservation law can be written as

$$\int_{\Omega_h} \frac{\partial \phi}{\partial t}\,\mathrm{d}\Omega_h + \int_{\Gamma_h} \boldsymbol{F}(\phi)\cdot\boldsymbol{n}\,\mathrm{d}\Gamma_h = 0, \tag{1.28}$$

which forms the basis for the finite-volume method. As an immediate consequence we have conservation of 'momentum' $\int_\Omega \phi\,\mathrm{d}\Omega$ as soon as along the outer boundary of the domain the flux function $\boldsymbol{F}$ vanishes, since all contributions of the fluxes along the interior cell faces cancel.

To investigate conservation of energy it is convenient first to discretize (1.28) further as

$$\boldsymbol{H}\frac{\mathrm{d}\boldsymbol{\phi}_h}{\mathrm{d}t} + \boldsymbol{M}_F\boldsymbol{\phi}_h = 0, \tag{1.29}$$

where $\boldsymbol{H}$ again is a diagonal matrix, now representing the size of the individual control volumes. The evolution of energy, defined by (1.27), then obeys

$$\frac{\mathrm{d}}{\mathrm{d}t}||\boldsymbol{\phi}_h||_h^2 = \frac{\mathrm{d}}{\mathrm{d}t}(\boldsymbol{\phi}_h^*\boldsymbol{H}\boldsymbol{\phi}_h) = -\boldsymbol{\phi}_h^*(\boldsymbol{M}_F + \boldsymbol{M}_F^*)\boldsymbol{\phi}_h.$$

Again it is observed that a positive real matrix $\boldsymbol{M}_F$ implies a decrease of energy (Theorem 1.5.3).

1.6.1 One space dimension

The convection-diffusion equation in one dimension fits in the general framework (1.28) with a flux function given by

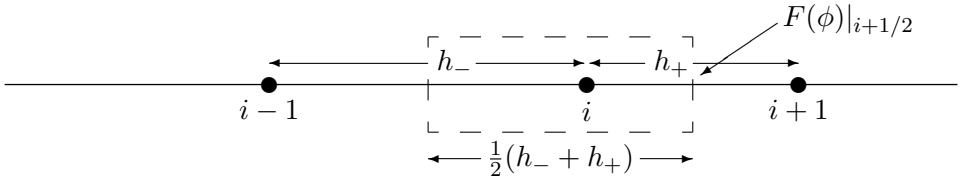
$$F(\phi) = u\,\phi - k\,\phi_x,$$

where for the moment u and k will be kept constant.

The choice of the control volumes is not straightforward. On uniform grids the control faces usually lie halfway the grid points, whereas at the same time the grid points lie in the centre of the control volumes. On non-uniform grids only one of these properties can hold. Below both variants will be treated, to show that their behaviour is completely different.

Faces halfway between grid points

We will start with a situation where the control faces lie halfway between the grid points; it is sometimes called a vertex-centered method (because in a rectangular setting the vertices of the control volumes lie central between the grid points).



In this situation the convective and diffusive fluxes are found immediately with second-order accuracy as (see notation in above figure)

$$F(\phi)|_{i+1/2} = u\frac{\phi_{i+1} + \phi_i}{2} - k\frac{\phi_{i+1} - \phi_i}{h_+}.$$

The discrete convection diffusion equation then becomes

$$\tfrac{1}{2}(h_- + h_+)\frac{\mathrm{d}\phi_i}{\mathrm{d}t} + \tfrac{1}{2}u(\phi_{i+1} - \phi_{i-1}) - k\left(\frac{\phi_{i+1} - \phi_i}{h_+} - \frac{\phi_{i-1} - \phi_i}{h_-}\right) = 0.$$

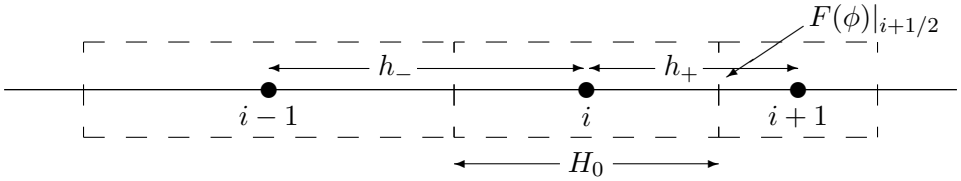
It is observed that this method equals Method A, and it might give the impression as if the finite-volume approach automatically results in a good discretization method. But this is not true in general, as we will see in the next variant.

Grid points halfway between faces

Next the cell-centered method in which the grid points lie halfway between the cell faces will be discussed. In this case the faces are not halfway the grid points, and a second-order approximation of the convective flux at a cell face becomes more complicated than in the previous approach.

Referring to the notation in the figure below, a linear interpolation results in a convective flux

$$F|_{i+1/2} = \frac{H_0}{2h_+}\phi_{i+1} + (1 - \frac{H_0}{2h_+})\phi_i.$$



For the diffusive flux one can do no better than in the previous approach, after which one ends up with

$$H_0 \frac{\mathrm{d}\phi_i}{\mathrm{d}t} + u \left\{ \frac{H_0}{2h_+}\phi_{i+1} + \left(\frac{H_0}{2h_-} - \frac{H_0}{2h_+} \right) \phi_i - \frac{H_0}{2h_-}\phi_{i-1} \right\} + \text{ diffusive terms} = 0.$$

We observe that the convective term contributes to the diagonal. Rewriting the terms between braces as an approximation for the derivative gives

$$\phi_x \approx \tfrac{1}{2} \frac{\phi_{i+1} - \phi_i}{h_+} + \tfrac{1}{2} \frac{\phi_i - \phi_{i-1}}{h_-}.$$

By comparison with the bottom lines in (1.18) and (1.19) if follows that, with respect to the convective term, this method is the 'average' of Methods A and B discussed earlier. On non-uniform grids it behaves just as bad as Method B, although for this 'averaged' method again Manteuffel and White (1986) have proven second-order convergence.

Because of this behaviour, Jameson *et al.* (1981) in their version of the cell-centered method have changed the computation of the convective flux into

$$F|_{i+1/2} = \tfrac{1}{2}(\phi_{i+1} + \phi_i).$$

This is a formula we have seen earlier, but as the cell faces are not halfway the grid points, its local truncation error is not second-order accurate. The resulting discretization of the convective derivative reads

$$\phi_x = \frac{\phi_{i+1} - \phi_{i-1}}{2H_0} + \tau_h,$$

where τ_h is the discretization error given by

$$\tau_h = (\frac{h_+ + h_-}{2H_0} - 1)\phi_x + \frac{h_+^2 - h_-^2}{2H_0}\phi_{xx} + \cdots .$$

Note that $2H_0$ is not the distance between the neighbouring grid points, therefore this method is not even consistent on arbitrary grids! In fact, also the discretization of the second-order derivative is inconsistent. On exponential grids with severe stretching this behaviour will show up (Veldman and Botta 1993). Then, why is Jameson's cell-centered method so popular? It is because its coefficient matrix can never become singular, just like that of Method A. Once more the favourable properties of the coefficient matrix make up for the unfavourable local truncation error.

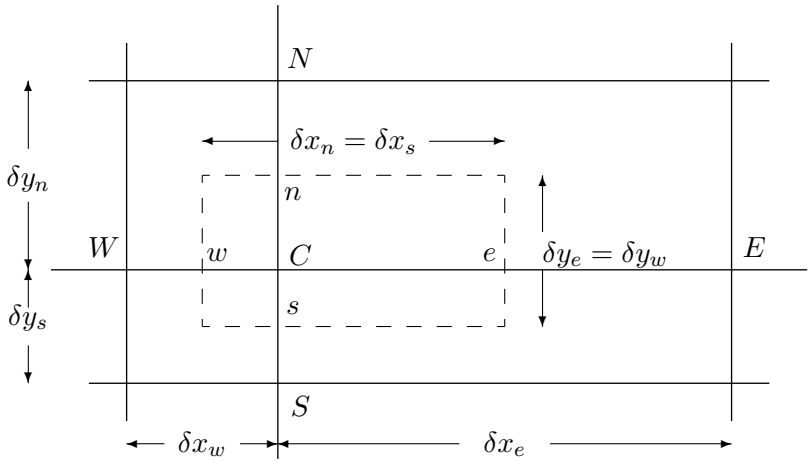
Another message to be learned from this section is that, although finite-volume discretization ensures conservation of a linear quantity like momentum, conservation of a quadratic quantity like energy is not automatic. In the previous section we have seen that the latter is a very crucial property.

1.6.2 More space dimensions

Next the vertex-centered method will be demonstrated in two dimensions. The steady convection-diffusion equation in conservation form reads

$$\int_\Gamma (\boldsymbol{u}\,\phi - k\,\mathrm{grad}\,\phi)\cdot\boldsymbol{n}\;\mathrm{d}\Gamma = 0, \tag{1.30}$$

in which $\boldsymbol{u}$ and k are allowed to vary over the domain of interest. It is assumed however that the velocity field $\boldsymbol{u} = (u, v)$ is divergence free, i.e. $\mathrm{div}\,\boldsymbol{u} = 0$.



In order not to complicate the presentation, we will restrict ourselves to a rectangular grid, but it is allowed to be non-uniform (for notation see the accompanying figure). Discretization of (1.30) yields in first instance

$$\begin{aligned}\delta y_e\left(u_e\phi_e - k_e\left.\frac{\partial\phi}{\partial x}\right|_e\right) + \delta x_n\left(v_n\phi_n - k_n\left.\frac{\partial\phi}{\partial y}\right|_n\right) + \\ \delta y_w\left(-u_w\phi_w + k_w\left.\frac{\partial\phi}{\partial x}\right|_w\right) + \delta x_s\left(-v_s\phi_s + k_s\left.\frac{\partial\phi}{\partial y}\right|_s\right) = 0.\end{aligned}$$

As the points $o,\ n,\ w$ and z lie halfway the grid points the fluxes are found with second-order accuracy leading to

$$\begin{aligned}\delta y_e\left(u_e\frac{\phi_E+\phi_C}{2} - k_e\frac{\phi_E-\phi_C}{\delta x_e}\right) + \delta x_n\left(v_n\frac{\phi_N+\phi_C}{2} - k_n\frac{\phi_N-\phi_C}{\delta y_n}\right) + \\ \delta y_w\left(-u_w\frac{\phi_W+\phi_C}{2} - k_w\frac{\phi_W-\phi_C}{\delta x_w}\right) + \delta x_s\left(-v_s\frac{\phi_S+\phi_C}{2} - k_s\frac{\phi_S-\phi_C}{\delta y_s}\right) = 0.\end{aligned} \tag{1.31}$$

If we assume that the continuity equation $\mathrm{div}\,u = 0$ has been discretized with the same control volume, then

$$\delta y_e u_e + \delta x_n v_n - \delta y_w u_w - \delta x_s v_s = 0. \tag{1.32}$$

Combining this with (1.31) it is observed that the contribution of ϕ_C from the convective term vanishes. We know already that this a favourable property. The following discrete equation

results

$$\delta y_e \left(\frac{u_e}{2} - \frac{k_e}{\delta x_e} \right) \phi_E + \delta x_n \left(\frac{v_n}{2} - \frac{k_n}{\delta y_n} \right) \phi_N + \delta y_w \left(-\frac{u_w}{2} - \frac{k_w}{\delta x_w} \right) \phi_W + \delta x_s \left(-\frac{v_s}{2} - \frac{k_s}{\delta y_s} \right) \phi_S + \left(\frac{\delta y_e}{\delta x_e} k_e + \frac{\delta x_n}{\delta y_n} k_n + \frac{\delta y_w}{\delta x_w} k_w + \frac{\delta x_s}{\delta y_s} k_s \right) \phi_C = 0.$$

We note that the coefficient of ϕ_E depends only on quantities defined in e. When the equation in the neighbouring control volume is constructed, ϕ_C in the neighbouring equation obtains the same coefficient, apart from a minus sign in the convective contribution. This makes the contribution of the convective terms skew symmetric, whereas the diffusive contribution is symmetric. And this is just what one wants!

Remark Generalization of the vertex-centered concept to general grids suggests choosing the control volumes according to a Voronoi tesselation.

1.7 Higher-order space discretization

Inspired by the success of the above second-order symmetry-preserving discretization, in Groningen we have concentrated on their extension to higher-order accuracy. As before, our choice of the discretization on non-uniform grids will be explained by studying the convection-diffusion equation. It will be discretized in a finite-volume fashion, as discussed already in Section 1.6, where the control faces are chosen halfway between the grid points (for notation see Figure 1.2).

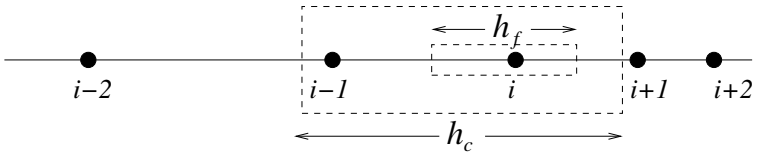


Figure 1.2: *Fine* (h_f) *and coarse* (h_c) *control volumes. The volume faces are located halfway the grid points* $(i, i \pm 1)$ *and* $(i, i \pm 2)$*, respectively.*

As in Section 1.6, with a second-order flux in $x_{i+1/2}$ given by

$$u \frac{\phi_{i+1} + \phi_i}{2} - k \frac{\phi_{i+1} - \phi_i}{x_{i+1} - x_i},$$

the semi-discrete convection-diffusion equation becomes

$$h_f \frac{\mathrm{d}\phi_i}{\mathrm{d}t} + \tfrac{1}{2} u (\phi_{i+1} - \phi_{i-1}) - k \left(\frac{\phi_{i+1} - \phi_i}{x_{i+1} - x_i} - \frac{\phi_{i-1} - \phi_i}{x_{i-1} - x_i} \right) = 0. \tag{1.33}$$

To turn this method into a fourth-order method, a similar equation on a two-times-larger control volume (see Figure 1.2) is written down

$$h_c \frac{\mathrm{d}\phi_i}{\mathrm{d}t} + \tfrac{1}{2} u (\phi_{i+2} - \phi_{i-2}) - \text{ diffusive terms } = 0. \tag{1.34}$$

The leading term in the discretization error can be removed through a Richardson extrapolation from (1.33) and (1.34). Since the errors in (1.33) and (1.34) are of third order (note that these discrete equations are multiplied by h as compared to their usual formulation), on a uniform grid this would mean to make a combination 8*Eq. (1.33)−Eq. (1.34). On a non-uniform grid one would be tempted to tune the weights to the actual mesh sizes, but we think it important that the skew symmetry of the convective contribution is maintained. This can only be achieved when the weights are taken independent of the grid location, and hence equal to the uniform weights. In this way the discretization of the convective derivative becomes

$$H_i \frac{\partial \phi}{\partial x} \approx \tfrac{1}{2}(-\phi_{i+2} + 8\phi_{i+1} - 8\phi_{i-1} + \phi_{i-2}), \tag{1.35}$$

where

$$H_i = 8h_f - h_c = \tfrac{1}{2}(-x_{i+2} + 8x_{i+1} - 8x_{i-1} + x_{i-2}).$$

On a uniform grid, of course, the usual fourth-order method is obtained, but on non-uniform grids the method differs considerably! In particular, the local truncation error given implicitly by

$$\begin{aligned} 2H_i \frac{\partial \phi}{\partial x} &= -\phi_{i+2} + 8\phi_{i+1} - 8\phi_{i-1} + \phi_{i-2} \\ &\quad + (h_{++}^2 - 8h_+^2 + 8h_-^2 - h_{--}^2)\phi_{xx} + \cdots \end{aligned}$$

(where $h_{++} = x_{i+2} - x_i$, $h_+ = x_{i+1} - x_i$, etc.) does not look very promising at first sight. On irregular grids it might even behave first-order (instead of fourth-order)!

The diffusive term undergoes a similar treatment leading to

$$H_i \frac{\partial^2 \phi}{\partial x^2} \approx 8\left(\frac{\phi_{i+1} - \phi_i}{x_{i+1} - x_i} - \frac{\phi_{i-1} - \phi_i}{x_{i-1} - x_i}\right) - \left(\frac{\phi_{i+2} - \phi_i}{x_{i+2} - x_i} - \frac{\phi_{i-2} - \phi_i}{x_{i-2} - x_i}\right). \tag{1.36}$$

Remark The expressions (1.35) and (1.36) can also be derived through a coordinate transformation $x = x(\xi)$ by writing

$$\frac{\mathrm{d}\phi}{\mathrm{d}x} = \frac{\mathrm{d}\phi}{\mathrm{d}\xi} \Big/ \frac{\mathrm{d}x}{\mathrm{d}\xi} \quad \text{and} \quad \frac{\mathrm{d}^2\phi}{\mathrm{d}x^2} = \frac{\mathrm{d}}{\mathrm{d}\xi}\left(\frac{\mathrm{d}\phi}{\mathrm{d}\xi} \Big/ \frac{\mathrm{d}x}{\mathrm{d}\xi}\right) \Big/ \frac{\mathrm{d}x}{\mathrm{d}\xi}.$$

Choose a uniform grid in ξ with mesh size Δ, then

$$\begin{aligned} \frac{\mathrm{d}\phi}{\mathrm{d}\xi} &= \frac{-\phi_{i+2} + 8\phi_{i+1} - 8\phi_{i-1} + \phi_{i-2}}{12\Delta} + O(\Delta^4), \\ \frac{\mathrm{d}x}{\mathrm{d}\xi} &= \frac{-x_{i+2} + 8x_{i+1} - 8x_{i-1} + x_{i-2}}{12\Delta} + O(\Delta^4) \\ \Rightarrow \quad \frac{\mathrm{d}\phi}{\mathrm{d}x} &= \frac{-\phi_{i+2} + 8\phi_{i+1} - 8\phi_{i-1} + \phi_{i-2}}{-x_{i+2} + 8x_{i+1} - 8x_{i-1} + x_{i-2}} + O(\Delta^4). \end{aligned}$$

Now, fourth-order behaviour looks obvious...

Testcase convection-diffusion

The performance of the above 2nd- and 4th-order methods will be demonstrated by comparing them with the traditional discretization methods based on Lagrange interpolation (minimising local truncation error). Since on uniform grids the methods are equal, an example with a boundary-layer character is chosen, requiring grid refinement near the outflow boundary $x = 1$. Thus, the steady convection-diffusion equation

$$\frac{\mathrm{d}\phi}{\mathrm{d}x} - k\frac{\mathrm{d}^2\phi}{\mathrm{d}x^2} = 0 \quad (0 < x < 1) \quad \phi(0) = 0,\ \phi(1) = 1 \tag{1.37}$$

is solved, with a diffusion coefficient equal to $k = 0.001$.

Two types of grid have been examined. Firstly, we will present results for piecewise-uniform grids where the interval $[0, 1]$ is split into two parts, in each of which the grid is taken uniform. The interface point, denoted by $1 - d$, is chosen near the edge of the boundary layer which has thickness $d = O(k)$. In both parts half of the grid points are positioned. Note that near the interface point the grid size abruptly changes, and only here the discretization differs from the Lagrangian one. Secondly, we have investigated more smoothly stretched grids. Results will be presented for a grid with a constant stretching (obtained through an exponential transformation function), again chosen such that half of the grid points lie in the boundary layer. Four discretization methods have been investigated:

- 2L: The traditional Lagrangian second-order method.
- 2S: The second-order symmetry-preserving method defined by (1.33).
- 4L: The traditional fourth-order Lagrangian method where we have implemented exact boundary conditions to circumvent the problem of a difference molecule that is too large near the boundary. In this way the boundary treatment does not interfere with the internal discretization.
- 4S: In the method defined by (1.35) and (1.36), the problems due to the 'missing' boundary conditions have been solved by using the second-order discretization (1.33) in the end points.

As the analytical solution to (1.37) is known, we can monitor the global discretization error defined by $||\phi_h - \phi_{\text{exact}}||_h$, where the norm is the kinetic energy norm defined in (1.27).

In Fig. 1.3 the global error is presented as a function of the mean mesh size ($= 1/N$, where N is the number of grid points). Two abrupt grids have been chosen, one in which $d = 5k$, which is too narrow to catch the boundary layer accurately, and one with $d = 15k$. Further, one exponential grid is shown with $d = 10k$ (on a smoothly stretched grid the location of the interface point is less critical).

A number of observations can be made. Firstly, often the fourth-order Lagrangian method is not more accurate than its second-order counterpart, especially when the number of grid points is not abundant. This explains why thus far fourth-order discretization has not been very popular.

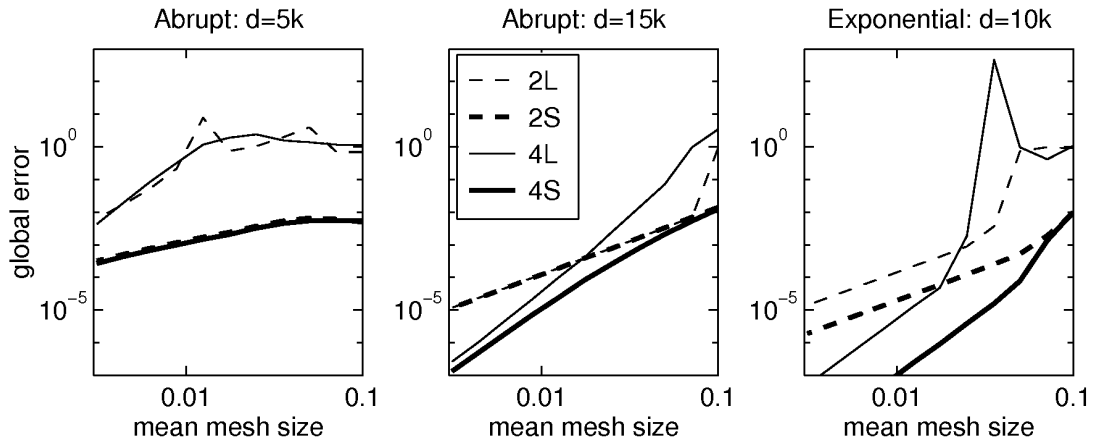


Figure 1.3: *The global error as a function of the mean mesh size. Half of the grid points is located in the boundary layer of thickness d. Four methods are shown: 2L (second-order Lagrangian), 2S (second-order symmetry-preserving), 4L (fourth-order Lagrangian with exact boundary conditions) and 4S (fourth-order symmetry-preserving with second-order boundary treatment).*

Secondly, when the number of grid points is low, Lagrangian discretization is much less accurate than symmetry-preserving discretization. Also, on the narrow abrupt grid Lagrangian discretization suffers most. In contrast, the symmetry-preserving discretization method behaves smoothly on all grids; even on the coarsest grids the error does not exceed 10^{-2}. Turbulent-flow simulations will always create the situation where one has to cope with limitations on the affordable number of grid points; hence methods that are less sensitive in this respect are preferable.

Thirdly, on the exponential grid the fourth-order Lagrangian method nearly breaks down for $N = 28$ where the stretching factor is 1.4 (which is large but not extreme). For this class of methods some eigenvalues of the coefficient matrix may be located in the left halfplane.

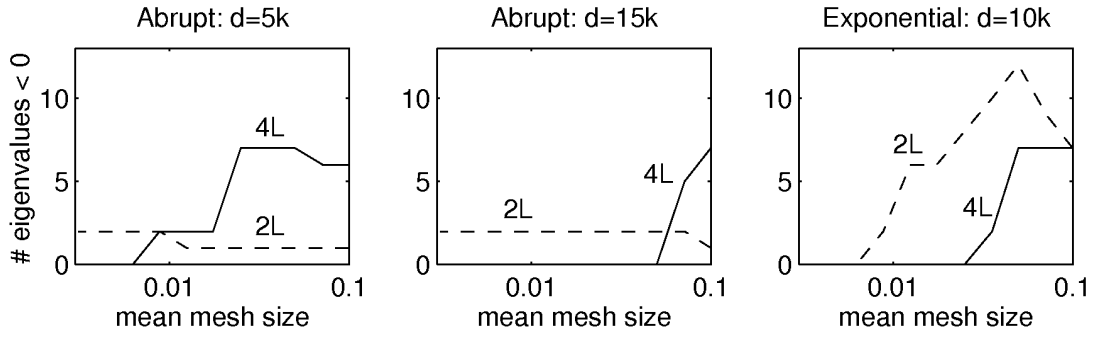


Figure 1.4: *The number of eigenvalues located in the (unstable) left halfplane for the Lagrangian methods as a function of mean mesh size (2L second-order; 4L fourth-order with exact boundary conditions).*

Upon refinement these eigenvalues will cross the imaginary axis, and, for this particular value of N, one of these eigenvalues almost vanishes, making the system almost singular. Note that the symmetry-preserving fourth-order method for this value of N is already very accurate, with an error around 10^{-5}.

When one or more eigenvalues of the coefficient matrix are located in the left halfplane, the semi-discrete system (1.26) is unstable, and cannot be integrated in the time domain; in the above examples the discrete solution has been obtained by a direct matrix solver. To illustrate how serious this problem is, in Fig. 1.4 the number of eigenvalues that are located in the unstable left halfplane are presented. Only the Lagrangian methods are shown, since the symmetry-preserving discretization always keeps the eigenvalues in the stable right halfplane. Note that upon grid refinement also all 'Lagrangian' eigenvalues will end up in the right halfplane, since eventually the mesh Péclet numbers will become smaller than 2.

1.8 Time integration

1.8.1 Stability analysis

Before proceeding towards more advanced space discretization methods for the convection-diffusion equation its time integration will be discussed, i.e. we start with the unsteady equation

$$\frac{\partial \phi}{\partial t} + u\,\frac{\partial \phi}{\partial x} - k\,\frac{\partial^2 \phi}{\partial x^2} = 0, \qquad 0 < x < 1. \tag{1.38}$$

This equation will be discretized explicitly in time and central in space, to obtain

$$\frac{\phi_j^{(n+1)} - \phi_j^{(n)}}{\delta t} + u\,\frac{\phi_{j+1}^{(n)} - \phi_{j-1}^{(n)}}{2h} - k\,\frac{\phi_{j+1}^{(n)} - 2\phi_j^{(n)} + \phi_{j-1}^{(n)}}{h^2} = 0.$$

After introduction of the following abbreviations

$$\eta \equiv \frac{u\,\delta t}{h} \quad (CFL\text{-number}) \text{ and } \quad d \equiv \frac{2k\,\delta t}{h^2} \tag{1.39}$$

the discrete equation can be rewritten as

$$\phi_j^{(n+1)} - \phi_j^{(n)} + \frac{\eta}{2}\left(\phi_{j+1}^{(n)} - \phi_{j-1}^{(n)}\right) - \frac{d}{2}\left(\phi_{j+1}^{(n)} - 2\phi_j^{(n)} + \phi_{j-1}^{(n)}\right) = 0. \tag{1.40}$$

In case of periodic boundary conditions

$$\phi(0,t) = \phi(1,t),$$

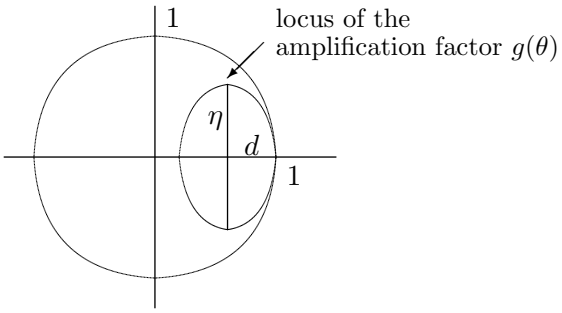
Fourier analysis yields the correct amplification behaviour of the time-integration method (see Appendix A.3). Thereto $\phi_j^{(n)} = c_\theta^{(n)}\, e^{ij\theta}$ is substituted, to find (after division by $e^{ij\theta}$)

$$\begin{aligned} c_\theta^{(n+1)} &= c_\theta^{(n)}\left[1 + \frac{d}{2}\left(e^{i\theta} - 2 + e^{-i\theta}\right) - \frac{\eta}{2}\left(e^{i\theta} - e^{-i\theta}\right)\right] \\ &= c_\theta^{(n)}\left[1 + d\,(\cos\,\theta - 1) - i\eta\,\sin\,\theta\right]. \end{aligned}$$

The expression between the brackets is the Fourier amplification factor

$$g(\theta) = 1 - d\,(1 - \cos\,\theta) - i\,\eta\,\sin\,\theta. \qquad (1.41)$$

In the complex plane it represents an ellipse with center $(1-d, 0)$ and half main axes equal to d and η, respectively.



The requirement $|g(\theta)| \leq 1$ leads to

$$\begin{aligned} & [1 + d\,(\cos\,\theta - 1)]^2 + \eta^2 \sin^2\theta \leq 1 \\ \Leftrightarrow \quad & d^2(\cos\,\theta - 1)^2 + 2d\,(\cos\,\theta - 1) + \eta^2(1 - \cos^2\theta) \leq 0 \\ \Leftrightarrow \quad & d^2(1 - \cos\,\theta) - 2d + \eta^2(1 + \cos\,\theta) \leq 0 \\ \Leftrightarrow \quad & d^2 + \eta^2 - 2d + (\eta^2 - d^2)\cos\,\theta \leq 0. \end{aligned} \qquad (1.42)$$

This has to hold for all θ. A simple observation shows that the left-hand side in (1.42) attains its maximum in either $\theta = 0$ or $\theta = \pi$. These two values lead to the necessary and sufficient conditions for Fourier stability

$$\eta^2 \leq d \;\text{ and }\; 0 \leq d \leq 1. \qquad (1.43)$$

The first condition in (1.43) can be rewritten as

$$\eta^2 \leq d \quad \Leftrightarrow \quad u^2\delta t \leq 2k \quad \Leftrightarrow \quad P \equiv \frac{2\eta}{d} \leq \frac{2}{\sqrt{d}}. \qquad (1.44)$$

We investigate zero-stability (see Appendix A.3) for $\delta t \to 0$ and $h \to 0$ with d constant. Taking the modulus of (1.41) yields

$$|g|^2 = \{1 - d\,(1 - \cos\,\theta)\}^2 + \eta^2\,\sin^2\theta = \{1 - d\,(1 - \cos\,\theta)\}^2 + \frac{u^2\delta t^2}{h^2}\,\sin^2\theta. \qquad (1.45)$$

The first term (between braces) in the right-hand side is less or equal unity as soon as $0 \leq d \leq 1$. The second term looks small enough anyway, it is even $O(\delta t^2)$, but the factor $1/h^2$ is misleading. It can become arbitrary large when $h \to 0$. Therefore it has to be 'tamed' by one of the factors δt; i.e. $\delta t/h^2$ should remain bounded. The latter is obviously satisfied when d is bounded, and the desired $O(\delta t)$ is achieved. Thus, ultimately we find that the scheme is zero-stable when

$$0 \leq d \leq 1, \qquad (1.46)$$

which is a weaker requirement than that for Fourier stability in (1.43). However, the coefficient of the δt-term in (1.45) can be large, and herewith $|g|$. This is a situation which is not acceptable in practice (as is shown below), and where one uses preferably the stronger concept of practical Fourier stability (see Appendix A.3).

It is interesting to analyse which of the conditions in (1.43) puts the strongest restriction on the time step. The diffusive condition $d \leq 1$ results in

$$\delta t \leq \frac{h^2}{2k} \equiv \delta t_{\text{diff}},$$

whereas rewriting (1.44) gives

$$\delta t \leq \delta t_{\text{Four}} \equiv \frac{2k}{u^2} = \frac{h^2}{2k}\,\frac{4k^2}{u^2h^2} = \delta t_{\text{diff}}\,\frac{4}{P^2}.$$

It follows that for $P < 2$ the diffusive condition $d \leq 1$ is most restrictive, whereas for $P > 2$ the condition $\eta^2 \leq d$ is most restrictive. A similar comparison holds with the convective condition $\eta \leq 1$ which we would have encountered when upwind discretization would have been applied (see the example in Appendix A.3.4):

$$\eta \leq 1 \quad \Leftrightarrow \quad \delta t \leq \delta t_{\text{conv}} \equiv \frac{h}{u} = \frac{h^2}{2k}\,\frac{2k}{hu} = \delta t_{\text{diff}}\,\frac{2}{P}.$$

A final aspect to investigate is the positivity of the discrete operator. Hereto the coefficients of all neighbouring points (in space as well as in time) have to be considered. It easily follows that the discrete operator in our problem is a positive operator if and only if $d \leq 1 \ \wedge \ P \leq 2$: a requirement which is stronger than (1.43).

1.8.2 Practical example

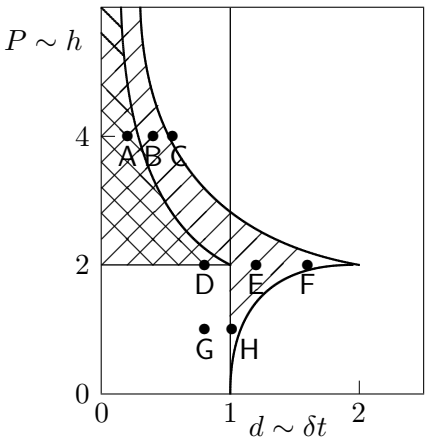
When mixed Dirichlet/Neumann boundary conditions

$$\phi(0,t) = \text{given}; \quad \frac{\partial \phi}{\partial x}(1,t) = \text{given}, \tag{1.47}$$

are applied, in Section 1.12 the matrix method predicts absolute stability under the conditions

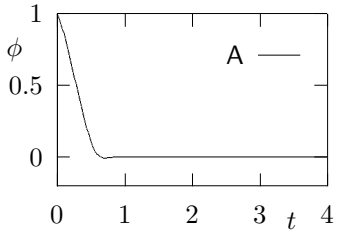
$$[\eta^2 \leq d^2 \quad \wedge \quad d + \sqrt{d^2 - \eta^2} \leq 2] \quad \vee \quad [d^2 \leq \eta^2 \leq 2d]. \tag{1.48}$$

In the adjacent figure the various stability regions have been indicated. In the 'white' area in the bottom-left corner the operator is positive; the discrete solution does not show any wiggles. The requirement of absolute stability for periodic boundary conditions allows the doubly shaded area top left. Here the time-integration method is practical (Fourier) stable. Zero-stability increases the allowable region to the strip $d \leq 1$. Also the Dirichlet–Neumann conditions increase the region of absolute stability; this time with the singly-shaded region given by (1.48).

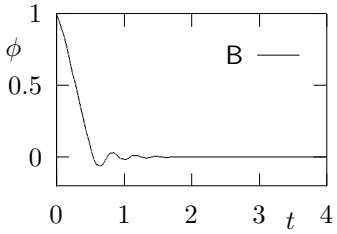


Next we will show what the discrete solution looks like for various choices of δt and h. The equation (1.38) is solved for $u = 2$ and $k = 0.05$, with homogeneous Dirichlet–Neumann boundary conditions (1.47). As initial condition $\phi(x,0) = x$ is chosen. Three choices for the grid size have been made: $h = 1/10$, $1/20$ and $1/40$. At each of these grid sizes the discrete solution has been determined for a number of choices for the time step δt. The various cases have been indicated in the figure with the letters A through H.

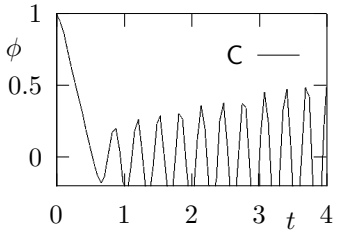
The behaviour of the discrete solution as a function of time is shown by plotting $\phi(1,t)$, i.e. the value at the Neumann boundary, for $0 \leq t \leq 4$.



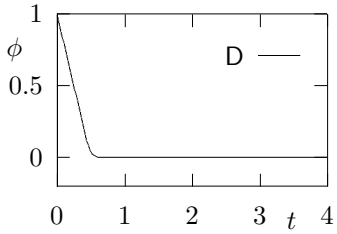
$A:\ h = 1/10;\ \delta t = 0.02$
$(A+\quad N+\quad P+)$



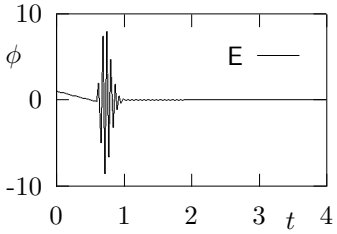
$B:\ h = 1/10;\ \delta t = 0.04$
$(A+\quad N+\quad P-)$



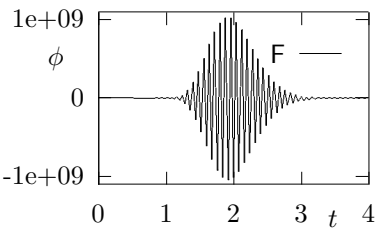
$C:\ h = 1/10;\ \delta t = 0.055$
$(A-\quad N+\quad P-)$



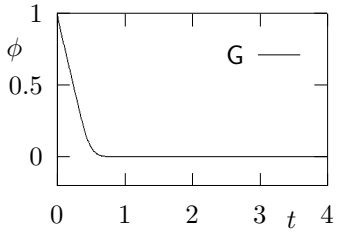
$D:\ h = 1/20;\ \delta t = 0.02$
$(A+\quad N+\quad P+)$



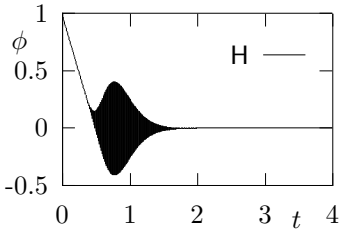
$E:\ h = 1/20;\ \delta t = 0.03$
$(A+\quad N-\quad P-)$



$F:\ h = 1/20;\ \delta t = 0.04$
$(A+\quad N-\quad P-)$



$G:\ h = 1/40;\ \delta t = 0.005$
$(A+\quad N+\quad P+)$



$H:\ h = 1/40;\ \delta t = 0.00655$
$(A+\quad N-\quad P-)$

Between parentheses + and − indicate whether for this choice of δt and h the time-integration method possesses absolute stability (A), zero-stability (N) and/or practical stability (P).

- Cases A, B and C possess a grid size $h = 1/10$. For A the time step has been selected such that the scheme is absolutely and practically stable for periodic boundary conditions. In B there is only absolute stability with D–N conditions, and in C there is no absolute stability. For each of these cases the scheme is zero-stable. All three solutions show oscillations (we are outside the area where the solution should be wiggle-free). Case A with $\delta t = 0.02$ looks quite nice, case C is unacceptable.

- In the cases D, E and F the grid size equals $h = 1/20$. In D a positive operator is created, which is absolutely, practically and zero-stable; the solution is monotone. E and F are only absolutely stable with D–N conditions. An instability develops, which after some time (when the Dirichlet condition becomes manifest) is being damped. But in the mean time the solution has been growing quite a bit (in $F : 10^9$). In linear-algebra terms, this (almost 'fatal') instability is due to (large) Jordan blocks which cause an algebraic growth; see the growth-estimate (A.16).

- In G and H the grid size equals $h = 1/40$. Case G satisfies all stability conditions and produces a nice solution. Case H is just outside the region of zero stability; this

stability boundary is at $\delta t = 0.00625$. Just as in cases E and F an instability develops which eventually gets damped.

The time step in case H is $\delta t = 0.00655$; this is a factor of three smaller than in cases A and D. With respect to accuracy the latter cases are quite good and the larger time step apparently is small enough. The fact that in cases H and G a much smaller time step has to be chosen is dictated by the stability of the method, rather than by its accuracy.

In conclusion it is remarked that the three best-looking solutions A, D and G precisely correspond with the cases where the time-integration method satisfies the criteria for practical stability, herewith illustrating the usefulness of this notion.

1.9 Burgers' equation – discontinuous solutions

In this section a first step is made towards solving the compressible Navier–Stokes equations. The main difference between the compressible and the incompressible equations is the possibility for discontinuities – shock waves – to develop in the solution. Using the one-dimensional equation of Burgers as a model, either in non-conservation form

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = 0, \tag{1.49}$$

or in conservation form

$$\frac{\partial u}{\partial t} + \frac{\partial}{\partial x}(\tfrac{1}{2}u^2) = 0, \tag{1.50}$$

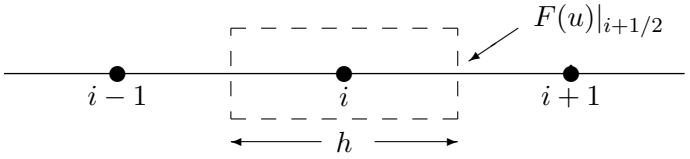
the performance of the thus far treated discrete approximation methods on non-smooth solutions will be demonstrated.

Finite-volume discretization

In Section 1.6 we have already encountered the finite-volume method (FVM) for the discretization of a conservation law like

$$\iint_\Omega \frac{\partial u}{\partial t}\,\mathrm{d}\Omega + \int_\Gamma \boldsymbol{F}(u)\cdot\boldsymbol{n}\,\mathrm{d}\Gamma = 0. \tag{1.51}$$

Consider this approach in a one-dimensional situation on a uniform grid as shown here:



The flux through the right-hand face of the control 'volume' around the point i is computed as

$$F(u)|_{i+1/2} := F(u_{i+1/2}) = F(\tfrac{1}{2}(u_i + u_{i+1})), \tag{1.52}$$

in which the value of u on this face is found through averaging the neighbouring cell values of u. For the discretization of the term $\partial F/\partial x$ in the differential equation corresponding to (1.51)

$$\frac{\partial u}{\partial t} + \operatorname{div} F(u) = 0, \tag{1.53}$$

this implies

$$\frac{\partial F}{\partial x} \approx \frac{F(u_{i+1/2}) - F(u_{i-1/2})}{h}. \tag{1.54}$$

As an alternative to (1.52), we could have averaged $F(u)|_{i+1/2}$ from neighbouring values of $F(u)$:

$$F(u)|_{i+1/2} := \tfrac{1}{2}\{F(u_i) + F(u_{i+1})\}. \tag{1.55}$$

This results in the discretization

$$\frac{\partial F}{\partial x} \approx \frac{F(u_{i+1}) - F(u_{i-1})}{2h}, \tag{1.56}$$

where a central finite-difference (FDM) discretization is recognised. In Section 2.3 it is demonstrated that the choice (1.52) is to be preferred, as in the Navier–Stokes equations it corresponds with (2.19′) which leads to a skew-symmetric discretization of convection.

An upwind-biased choice of the flux is given by $F(u)|_{i+1/2} := F(u_i)$ (for $u > 0$), leading to the upwind discretization

$$\frac{\partial F}{\partial x} \approx \frac{F(u_i) - F(u_{i-1})}{h}. \tag{1.57}$$

Burgers' equation

We will study Burgers' equation (named after Johannes Marinus Burgers, a Delft professor of fluid dynamics) in a situation with shock waves. In order to obtain the correct analytical shock solution it is necessary to employ the conservation form (1.50). It is known that the propagation speed S of a discontinuity in its solution is given by

$$-S[u] + [\tfrac{1}{2}u^2] = 0, \tag{1.58}$$

where $[\cdot]$ is the jump of the argument across the discontinuity. For Burgers' equation this results in

$$S = \tfrac{1}{2}(u_{\text{Left}} + u_{\text{Right}}). \tag{1.59}$$

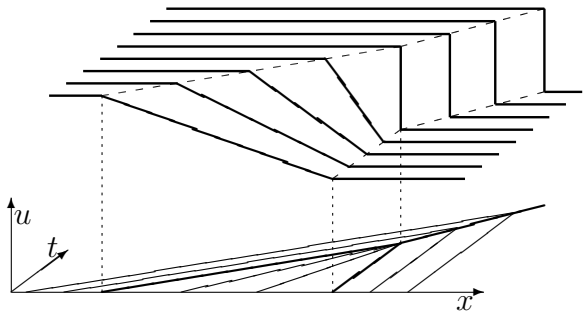
As initial condition we choose

$$x \le 0 :\ u = 1; \qquad 0 \le x \le 1 :\ u = 1 - x; \qquad 1 \le x :\ u = 0.$$

The characteristics of Burgers' equation, along which $u =$ Constant, have a slope $\mathrm{d}x/\mathrm{d}t = u$. As $u_{\text{Left}} > u_{\text{Right}}$ the waves from the left overtake the other ones and a shock wave is created, with a shock speed $S = \frac{1}{2}$ as follows from (1.59). The analytic solution is shown below.

Discretization

We will now solve Burgers' equation by several discretization methods, globally distinguished as *i)* upwind or *ii)* central (with or without explicitly added diffusion). Moreover, both the conservative form (1.50) and the non-conservative form (1.49) will be considered. Forward Euler is used as time integrator. For all computations the mesh size in x-direction has been chosen as $h = 1/10$ with a time step $\delta t = 1/40$; the CFL-number becomes $\eta \equiv u\,\delta t/h \le 1/4$. Since there is no diffusion, the mesh-Péclet number $P \equiv u\,h/k$ is infinite.



In the table below the various discretizations for the convective terms are summarised

	non-conservation form	conservation form
upwind FDM/FVM	$u_i \frac{u_i - u_{i-1}}{h}$	$\frac{\frac{1}{2}u_i^2 - \frac{1}{2}u_{i-1}^2}{h}$
central FDM	$u_i \frac{u_{i+1} - u_{i-1}}{2h}$	$\frac{\frac{1}{2}u_{i+1}^2 - \frac{1}{2}u_{i-1}^2}{2h}$
central FVM	not applicable	$\frac{\frac{1}{2}u_{i+1/2}^2 - \frac{1}{2}u_{i-1/2}^2}{h}$

Below it will be shown that the difference between both columns is essential. The difference between both central methods in the right-hand column is less important; both are a discrete conservation law, only the flux computation is slightly different – compare (1.52) with (1.55).

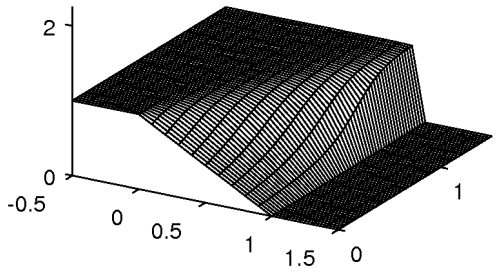
In Section 1.8 the stability of centrally-discretized convection-diffusion equations has been studied. Next to the CFL-number η, the quantity $d \equiv 2k\,\delta t/h^2$ has been introduced. It was demonstrated that forward Euler is zero-stable if and only if $d \leq 1$. Practical stability is achieved for (see (1.43)) $\eta^2 \leq d \leq 1$, whereas for Dirichlet/Neumann conditions the region with absolute stability is larger than that of practical stability. Without diffusion $d = 0$, hence central discretization is always zero-stable. At the same time, for no choice of $\delta t > 0$ the criterion $\eta^2 \leq d$ can be satisfied (unless $u = 0$), hence there is no practical stability.

Discrete solution

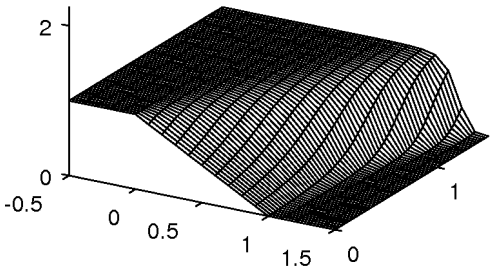
Let us first observe the difference between upwind discretization and central discretization. The upwind solution is smooth, as was to be expected from the theory of positive operators. The practical stability limit is $CFL = 1$ which is clearly satisfied.

Remark As u is not constant we cannot perform the 'accuracy trick for upwind' $CFL \equiv 1$ (see the example in Section A.3.4).

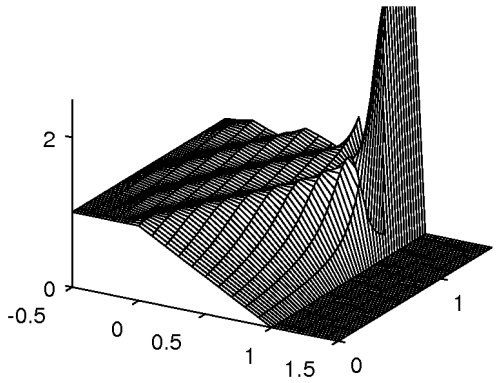
Clearly visible in the central solution are the strongly growing oscillations induced by the steep gradient. If we want to suppress these, numerical diffusion has to be added explicitly. Its influence is investigated with $k_a = 1/20$, chosen such that for $u \leq 1$ the mesh-Péclet number satisfies $P \leq 2$. Further it follows that $d = 1/4$, hence the time integration is stable under all definitions (we are in the blank bottom-left area in the figure on page 29). As expected theoretically, all oscillations have disappeared, but the discontinuity has been smeared out excessively. Note that in regions with low values of u, this amount of viscosity is much



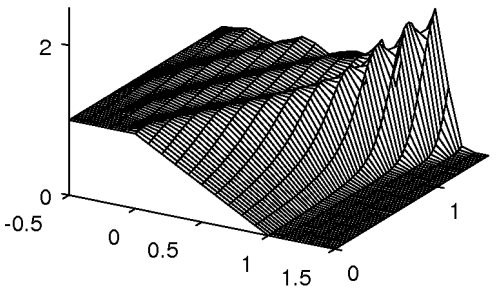
upwind FDM
non-conservation form



upwind FDM/FVM
conservation form



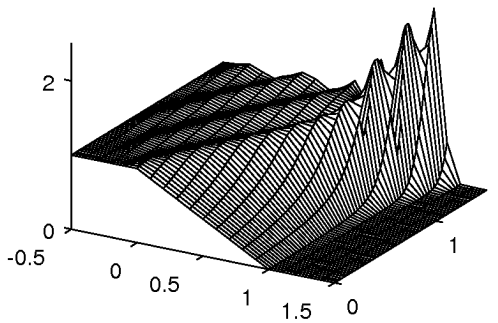
central FDM
non-conservation form



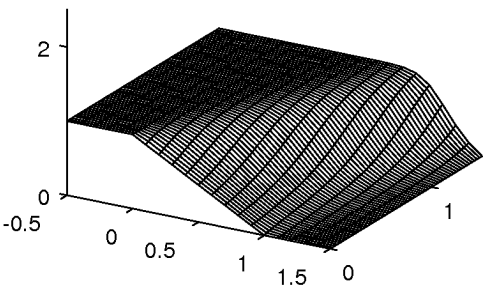
central FDM
conservation form

larger than the upwind viscosity (which scales with u). In this respect, upwind isn't that bad...

Next, observe the propagation speed of the shock. In both solutions of the non-conservation form (1.49) the shock does not move; the solutions of the conservation form do have a proper shock speed. This holds for both upwind and central solutions. It is concluded that analytic jump conditions, like (1.58), only correctly carry over to the discrete form if we start from the conservation form of the equations; in fact we should start from the conservation law (1.51).



central FVM



central FVM with artificial
diffusion such that P < 2

Since in the finite-volume discretization one is obliged to start from the conservation

law, the propagation of discontinuities 'automatically' proceeds in the correct way. With a finite-difference or finite-element discretization you have to remind yourself to start with the conservative form, and than it still can go wrong. As an example, the convection-diffusion equation earlier in this chapter is formulated in conservation form (simply because it is linear with constant coefficients). However, its non-uniform discretization (1.19) is not a discrete conservation law, whereas the variant (1.18) is! Hence, with regard to discontinuous solutions, the finite-volume discretization is essentially different from the other discretization methods.

The above figures clearly show the influence of using the conservation form or not (left: no, right: yes). The figure below shows the central FVM (by definition in conservation form) with and without adding numerical diffusion. It will be clear that the discrete solutions for discontinuous solutions are not yet really convincing. Therefore special methods have been developed, e.g. based on the concept of (nonlinear) limiters, to reduce the amount of oscillations in the solution. The monograph by Hirsch (1990) gives a nice introduction into this subject.

1.10 Nonlinear schemes and flux-limiting

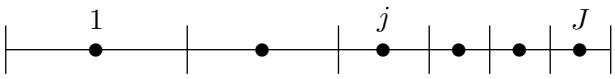
In many applications in fluid dynamics the Péclet/Reynolds number satisfies $Pe, Re \gg 1$, and as a consequence the diffusion terms are small in large parts of the domain. It is therefore especially important to approximate the convection term accurately, the more so because in inviscid flow models, such as the Euler equations, convection terms are the only ones present. When improving the accuracy of the discretization of the convection terms, the order barrier on schemes of positive type makes itself felt. In order to obtain higher order accuracy without spurious oscillations a way around this barrier has to be found. This leads to the introduction of *flux-limiting*[7] .

The κ-scheme

Let us reconsider equation

$$\frac{\mathrm{d}u\varphi}{\mathrm{d}x} - \frac{\mathrm{d}}{\mathrm{d}x}(k\frac{\mathrm{d}}{\mathrm{d}x}) = q, \quad x \in (0,1),$$

discretized with the finite volume method on the cell-centered grid



with grid size H_j, according to

$$F_{j+1/2} - F_{j-1/2} = H_j q_j, \quad j = 1, 2, \cdots, J.$$

Let us split the flux function F in its convective and diffusive parts:

$$F = F^c + F^v, \quad F^c = u\varphi, \quad F^v = -k\,\mathrm{d}\varphi/\mathrm{d}x.$$

As compared to the second order central scheme or the first order upwind scheme, with the κ-scheme one tries to enhance the accuracy by using an additional cell-center value, namely

[7] This section is based on Wesseling's notes for the JMBC course CFD 1.

φ_{j-1} for $u_{j+1/2} \geq 0$ or φ_{j+2} for $u_{j+1/2} < 0$. In order to maintain second order accuracy, to the convective flux function of the central second order scheme a higher order term is added, as follows:

$$F^c_{j+1/2} \cong \begin{cases} u_{j+1/2}\{\frac{1}{2}(\varphi_j + \varphi_{j+1}) + \frac{1-\kappa}{4}(-\varphi_{j-1} + 2\varphi_j - \varphi_{j+1})\}, & u_{j+1/2} \geq 0 \\ u_{j+1/2}\{\frac{1}{2}(\varphi_j + \varphi_{j+1}) + \frac{1-\kappa}{4}(-\varphi_j + 2\varphi_{j+1} - \varphi_{j+2})\}, & u_{j+1/2} < 0 \end{cases} \tag{1.60}$$

The approximation to the convection term becomes

$$\left(\frac{\mathrm{d}u\varphi}{\mathrm{d}x}\right)_j \cong \frac{1}{h_j}(F^c_{j+1/2} - F^c_{j-1/2}) \tag{1.61}$$

In the 'compressible world' this one-parameter family of schemes is called the κ-scheme (Van Leer, 1977a). If $u =$ constant > 0 the stencil is given by

$$\frac{u}{h_j}[\frac{1-\kappa}{4} \quad \frac{3\kappa-5}{4} \quad \frac{3-3\kappa}{4} \quad \frac{1+\kappa}{4} \quad 0]$$

It is equal to the family of λ-schemes from the 'incompressible world', presented in Section 1.3, through the relation $\lambda = (1-\kappa)/4$. Hence, for $\kappa = -1$ we obtain the one-sided second-order upwind scheme B3 (e.g. Warming and Beam 1976). With $\kappa = 0$ one obtains Fromm's zero average phase error scheme (Fromm 1968). In this scheme the terms quadratic in the Courant number are neglected, which are meant to improve time accuracy, of no concern here; this scheme results from optimizing, among 5-point schemes, for the propagation of a step function over one time step in the absence of diffusion (Wesseling 1973). For $\kappa = 1/3$ the third order upwind biased scheme (Anderson et al. 1985) results. For $\kappa = 1/2$ we have the QUICK scheme proposed by Leonard (1979a). Finally, $\kappa = 1$ gives the central scheme (1.4).

A remark on the local truncation error

The global and local truncation errors in grid point j have been defined in the preceding chapter as

$$e_j \equiv \varphi(x_j) - \varphi_j, \quad \tau_j \equiv L_h e_j$$

with $\varphi(x)$ the exact solution and L_h the discrete operator. It is assumed that L_h is scaled such that it approximates the differential operator. Let $k = 0$, and assume $\tau_j = O(h^p)$ and $e_j = O(h^q)$. Since (1.61) is an exact consequence of the differential equation and contains no error, it may come as a surprise that q is not maximized by making $F^c_{j\pm1/2}$ as accurate as possible, which is the case for $\kappa = 1/2$, resulting in $p = q = 2$. Instead, q is maximized for $\kappa = 1/3$, resulting in $p = q = 3$. The underlying cause of this apparent discrepancy is, that (1.61) is not *equivalent* to the differential equation. It follows from the differential equation, but the differential equation does not follow from (1.61). The situation is illustrated in exercise 1.10.1.

The fourth order central scheme

Again using a 5-point stencil, but no upwind bias, $\tau_j = O(h^4)$ on a uniform grid is obtained by

$$F^c_{j+1/2} \cong u_{j+1/2}\{\frac{7}{12}(\varphi_j + \varphi_{j+1}) - \frac{1}{12}(\varphi_{j-1} + \varphi_{j+2})\}$$

If $u = constant$ the stencil for the resulting approximation of $(\mathrm{d}u\varphi/\mathrm{d}x)_j$ is given by

$$\frac{u}{12h}[1 \quad -8 \quad 0 \quad 8 \quad -1] \tag{1.62}$$

Approximation of $\varphi_{j+1/2}$ by interpolation to highest order gives a different stencil, with $\tau_j = O(h^3)$, cf. exercise 1.10.1.

Nonlinear schemes

Because the order of accuracy of the preceding schemes is higher than one, the order barrier theorem predicts that these schemes are not of positive type. This is immediately confirmed by a look at their stencils. Hence, the maximum principle does not apply, and spurious wiggles may occur. Of course it is desirable to combine higher order accuracy with the positive type property, in order to avoid spurious wiggles. To achieve this, a way around the order barrier theorem has to be found. This may be done by using *nonlinear schemes*, i.e. schemes that are nonlinear, even though the differential equation that is approximated is linear. A great variety of nonlinear additions to linear schemes has been proposed in the pursuit of monotonicity. Many of these can be formulated in a unifying framework laid out by Leonard (1988), called the normalized variable diagram, which we will first discuss. A different but equivalent way to formulate nonlinear schemes of positive type is provided by the frequently used concept of flux limiting. This will also be discussed.

Normalized variable diagram

Point of departure is an upwind biased linear scheme, for which we take the κ-scheme (1.60). For brevity we assume $u_j \geq 0$; the contrary case may be handled by symmetry. In Leonard (1988) the following *normalized variable* is introduced:

$$\tilde{\varphi} = \frac{\varphi - \varphi_{j-1}}{\varphi_{j+1} - \varphi_{j-1}} \tag{1.63}$$

From (1.60) it follows that for the κ-scheme we have

$$\tilde{\varphi}_{j+1/2} = \frac{2-\kappa}{2}\tilde{\varphi}_j + \frac{1+\kappa}{4} \tag{1.64}$$

The graph of $\tilde{\varphi}_{j+1/2}$ versus $\tilde{\varphi}_j$ is called by Leonard (1988) the *characteristic* of the scheme and can be plotted in the *normalized variable diagram*, see Fig. 1.5 for example. A general nonlinear scheme is defined by

$$\tilde{\varphi}_{j+1/2} = f(\tilde{\varphi}_j) \tag{1.65}$$

For the κ-scheme, f is linear.

The order of accuracy of the nonlinear scheme defined by (1.65) can be studied as follows. Let $u =$ constant. Then $\varphi_{j+1/2} - \varphi_{j-1/2}$ has to approximate $h_j(\mathrm{d}\varphi/\mathrm{d}x)_j$. Let the grid be uniform. In order to be able to analyze the local truncation error by Taylor expansion, f is approximated locally by, assuming that $f''(1/2)$ is bounded,

$$f(\tilde{\varphi}_j) \cong f(\tfrac{1}{2}) + (\tilde{\varphi}_j - \tfrac{1}{2})f'(\tfrac{1}{2}) + O(\tilde{\varphi}_j - \tfrac{1}{2})^2 \tag{1.66}$$

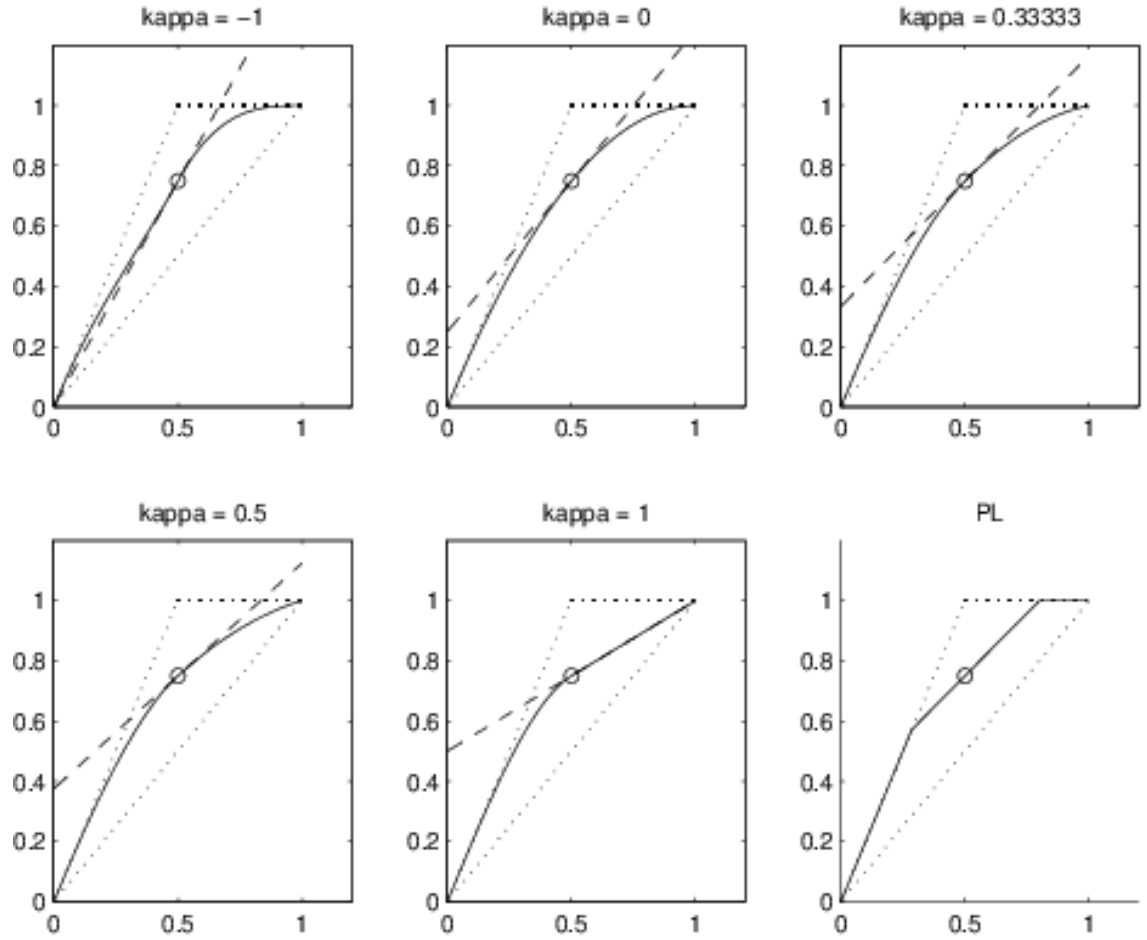


Figure 1.5: *Normalized variable diagram:* $f = f(\tilde{\varphi})$. *o : the point* $P = (1/2, 3/4)$*; - - - : κ-scheme; — :* $f(\tilde{\varphi})$.

One finds

$$\begin{aligned} \varphi_{j+1/2} - \varphi_{j-1/2} &= h(\mathrm{d}\varphi/\mathrm{d}x)_j + h^2(\mathrm{d}^2\varphi/\mathrm{d}x^2)_j\{2f(\tfrac{1}{2}) - \tfrac{3}{2}\} \\ &\quad + h^3(\mathrm{d}^3\varphi/\mathrm{d}x^3)_j\{\tfrac{7}{6} - f(\tfrac{1}{2}) - \tfrac{1}{2}f'(\tfrac{1}{2})\} + O(h^4) \end{aligned} \tag{1.67}$$

We see that the scheme is first order accurate for arbitrary $f(\frac{1}{2})$, which justifies the choice made for the normalized variables, and the decision to develop $f(\tilde{\varphi}_j)$ arround $\tilde{\varphi}_j = 1/2$ in (1.66). If

$$f(1/2) = 3/4 \tag{1.68}$$

the scheme is second order. As expected, the κ-scheme satisfies (1.68) for all κ. Finally, if

$$f'(1/2) = 5/6 \tag{1.69}$$

the scheme is third order. As expected, the $\kappa = 1/3$ scheme satisfies (1.69). In other words, the characteristics of second order schemes pass through the point $P = (1/2, 3/4)$ in the normalized variable diagram (fig. 1.5), and for third order schemes are tangent in P to the

characteristic of the $\kappa = 1/3$ scheme.

Next, we derive further conditions to be satisfied by f. In order for the scheme to exclude spurious wiggles and to be more than first order accurate, f has to be nonlinear. We consider a number of possibilities for the values of φ_{j-1}, φ_j and φ_{j+1}, see fig. 1.6. If $\varphi_j \in [\varphi_{j-1}, \varphi_{j+1}]$ (i.e. $0 \le \tilde{\varphi}_j \le 1$) the distribution of values is monotone, and monotonicity is preserved if we

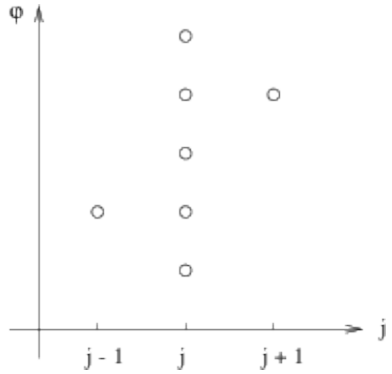


Figure 1.6: *Various values of φ_j in comparison to φ_{j-1} and φ_{j+1}.*

choose $\varphi_{j+1/2} \in [\varphi_j, \varphi_{j+1}]$. For $f(\tilde{\varphi}_j) = \tilde{\varphi}_{j+1/2}$ this leads to

$$\tilde{\varphi}_j \le f(\tilde{\varphi}_j) \le 1, \quad 0 \le \tilde{\varphi}_j \le 1 \tag{1.70}$$

If $\varphi_j \ne [\varphi_{j-1}, \varphi_{j+1}]$ the distribution of values is not monotone, and in order to steer the solution to monotonicity the first order upwind scheme $\varphi_{j+1/2} = \varphi_j$ is applied, resulting in

$$f(\tilde{\varphi}_j) = \tilde{\varphi}_j, \quad \tilde{\varphi}_j \notin [0, 1] \tag{1.71}$$

Finally, we remark that in the neglected $O(h^4)$ term in (1.67) there are terms proportional to f'', so that it seems advisable to keep f'' bounded. Examples of nonlinear characteristics satisfying these criteria are given in Fig. 1.5. In the origin and in (1,1) f'' is not bounded (since here the characteristic is continued with slope 1 according to (1.71)), but this is allowed, since here the scheme switches to first order upwind, so that lower order terms dominate in the expansion (1.67).

The flux limiter

A different way to formulate nonlinear schemes is as follows. In order to choose the flux $F^c_{j+1/2} = u_{j+1/2}\,\varphi_{j+1/2}$ we may also write, instead of (1.65),

$$\varphi_{j+1/2} = \varphi_j + \frac{1}{2}\psi(r_j)(\varphi_{j+1} - \varphi_j) \tag{1.72}$$

Van Leer (1974) has proposed to make ψ a function of the ratio of consecutive gradients:

$$r_j = \frac{\varphi_j - \varphi_{j-1}}{\varphi_{j+1} - \varphi_j}$$

The function $\psi(r)$ determines a correction on the upwind flux $u_{j+1/2}\,\varphi_j$, and is called the *flux limiter*. A survey of flux limiters (in the instationary case) is given by Sweby (1984). The

relation between the flux limiter ψ and the normalized variable function is as follows. We have

$$r_j = \frac{\tilde{\varphi}_j}{1-\tilde{\varphi}_j}, \quad \tilde{\varphi}_j = \frac{r_j}{1+r_j} \tag{1.73}$$

Equation (1.72) can be rewritten as

$$\tilde{\varphi}_{j+1/2} = \tilde{\varphi}_j + \tfrac{1}{2}\psi(r_j)(1-\tilde{\varphi}_j)$$

Comparison with (1.65) and substitution of (1.73) gives the following relation, suppressing indices:

$$f(\tilde{\varphi}) = \{r + \tfrac{1}{2}\psi(r)\}/(1+r), \quad \psi(r) = 2\{f(\tilde{\varphi}) - \tilde{\varphi}\}/(1-\tilde{\varphi}) \tag{1.74}$$

From (1.74) follows that if f satisfies (1.70) and (1.71), then ψ satisfies, deleting subscripts for brevity,

$$0 \le \psi(r) \le 2\,, \quad 0 \le r \le \infty\,, \quad \psi(r) = 0\,, \quad r < 0 \tag{1.75}$$

Flux-limited schemes of positive type

We will derive conditions for flux-limited schemes to be of positive type. Let us consider the general case where $u(x)$ may change sign. The scheme for the convection term is written as

$$L_h\varphi_j = \frac{1}{h_j}F^c|_{j-1/2}^{j+1/2}\,, \quad F^c_{j+1/2} = (u\varphi)_{j+1/2}\,. \tag{1.76}$$

If $u_{j+1/2} \ge 0$, $\varphi_{j+1/2}$ is given by (1.72). If $u_{j+1/2} < 0$, upwind bias is applied in the other direction. By symmetry we obtain:

$$\varphi_{j+1/2} = \varphi_{j+1} + \tfrac{1}{2}\psi(\frac{1}{r_{j+1}})(\varphi_j - \varphi_{j+1})\,, \quad u_{j+1/2} < 0\,. \tag{1.77}$$

The discretization in the case where $u_{j+1/2}$ is arbitrary can now be written as

$$\begin{aligned}
&F^c_{j+1/2} = F^+_{j+1/2} + F^-_{j+1/2}\,, \quad F^{\pm} = (u\varphi)_{j+1/2}\,,\\
&\varphi^+_{j+1/2} = \varphi_j + \tfrac{1}{2}\psi(r_j)(\varphi_{j+1} - \varphi_j)\,, \quad u_{j+1/2} \ge 0\,, \quad F^+_{j+1/2} = 0\,, \quad u_{j+1/2} < 0\,,\\
&\varphi^-_{j+1/2} = \varphi_{j+1} + \tfrac{1}{2}\psi(\frac{1}{r_{j+1}})(\varphi_j - \varphi_{j+1})\,, \quad u_{j+1/2} < 0\,, \quad F^-_{j+1/2} = 0\,, \quad u_{j+1/2} \ge 0\,,\\
&L_h\varphi_j = \frac{1}{h_j}(F^+ + F^-)|_{j-1/2}^{j+1/2}\,.
\end{aligned} \tag{1.78}$$

In order to derive conditions on the limiter $\psi(r)$ for scheme (1.78) to be of positive type we write it in the following form:

$$L_h\varphi_j = \{a_{j+1/2}(\varphi_{j+1} - \varphi_j) + b_{j-1/2}(\varphi_{j-1} - \varphi_j)\}/h_j\,, \tag{1.79}$$

where $a_{j+1/2}$ and $b_{j-1/2}$ are functions of $\varphi_j, \varphi_{j\pm1}, \varphi_{j\pm2}, \ldots$. We can write

$$\begin{aligned}
a_{j+1/2} &= \frac{F^-_{j+1/2} - F^-_{j-1/2}}{\varphi^-_{j+1/2} - \varphi^-_{j-1/2}}\,\frac{\varphi^-_{j+1/2} - \varphi^-_{j-1/2}}{\varphi_{j+1} - \varphi_j},\\
b_{j-1/2} &= \frac{F^+_{j+1/2} - F^+_{j-1/2}}{\varphi^+_{j+1/2} - \varphi^+_{j-1/2}}\,\frac{\varphi^+_{j+1/2} - \varphi^+_{j-1/2}}{\varphi_{j-1} - \varphi_j}.
\end{aligned} \tag{1.80}$$

Note that $\mathrm{d}F^-/\mathrm{d}\varphi^- \leq 0$ and $\mathrm{d}F^+/\mathrm{d}\varphi^+ \geq 0$, so that by the mean value theorem we have

$$\frac{F^-_{j+1/2} - F^-_{j-1/2}}{\varphi^-_{j+1/2} - \varphi^-_{j-1/2}} \leq 0 \ , \quad \frac{F^+_{j+1/2} - F^+_{j-1/2}}{\varphi^+_{j+1/2} - \varphi^+_{j-1/2}} \geq 0 \ . \tag{1.81}$$

Comparison of (1.79) with the definition of positivity (see Appendix A.1) shows that for the scheme to be of positive type it is necessary and sufficient that

$$a_{j+1/2} \leq 0 \ , \quad b_{j-1/2} \leq 0 \ . \tag{1.82}$$

It follows that we must have

$$\frac{\varphi^-_{j+1/2} - \varphi^-_{j-1/2}}{\varphi_{j+1} - \varphi_j} \geq 0 \ , \quad \frac{\varphi^+_{j+1/2} - \varphi^+_{j-1/2}}{\varphi_{j-1} - \varphi_j} \leq 0 \ . \tag{1.83}$$

Using (1.72) and (1.77) we see that we must have

$$2 - \psi(r) + \frac{\psi(s)}{s} \geq 0 \ , \quad \forall r, s \in \mathbb{R} \tag{1.84}$$

This is necessary and sufficient. To have $a_{j+1/2}$ and $b_{j+1/2}$ bounded we must also have

$$\frac{\varphi^-_{j+1/2} - \varphi^-_{j-1/2}}{\varphi_{j+1} - \varphi_j} \leq M \ , \quad \frac{\varphi^+_{j+1/2} - \varphi^+_{j-1/2}}{\varphi_{j-1} - \varphi_j} \geq -M \ ,$$

for some $M > 0$. This gives

$$2 - \psi(r) + \frac{\psi(s)}{s} \leq 2M \ , \quad \forall r, s \in \mathbb{R} \tag{1.85}$$

Combination of (1.84) and (1.85) gives

$$-2 \leq -\psi(r) + \frac{\psi(s)}{s} \leq 2M - 2 \ , \quad \forall r, s \in \mathbb{R} \tag{1.86}$$

If we require

$$1 - M \leq \psi(r) \leq m \leq 2 \tag{1.87}$$

then equation (1.86) gives

$$m - 2 \leq \frac{\psi(s)}{s} \leq M - 1 \ . \tag{1.88}$$

Equations (1.87) and (1.88) imply that the graph of $\psi(r)$ must lie in the shaded region of Fig. 1.7. For $m = 2, M = 3$ we obtain the first order TVD region derived by Sweby (1984) for the instationary case.

The aim of flux-limiting is, of course, to obtain schemes of positive type of second (or higher) order of accuracy. For second order accuracy we must satisfy equation (1.70). Using (1.73) and (1.74) this gives

$$\psi(1) = 1 \tag{1.89}$$

To have this point in the admissible region we must take $M \geq 2$. To make sure that the graph of $\psi(r)$ passes through $\psi(1) = 1$ we may formulate the following additional requirement:

$$1 \geq \psi(r) \geq r \ , \quad 1 \leq \psi(r) \leq r \ . \tag{1.90}$$

This results in the admissible region (shaded) of Fig. 1.8, which corresponds to the second order TVD region of Sweby (1984) if $M = 3, m = 2$.

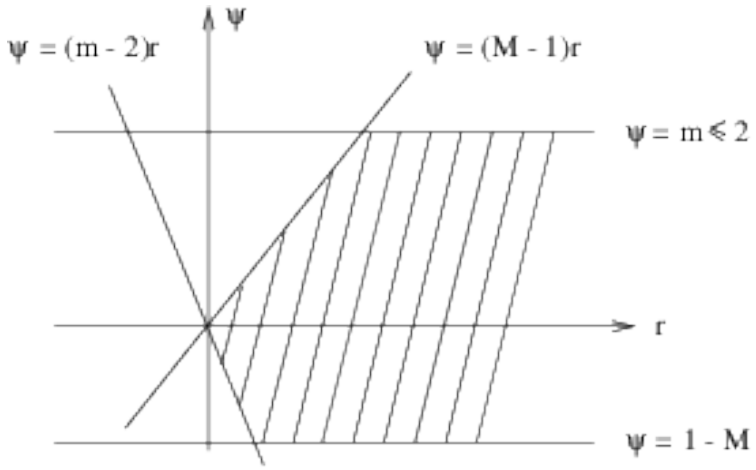


Figure 1.7: *Admissible region for graph of $\psi(r)$.*

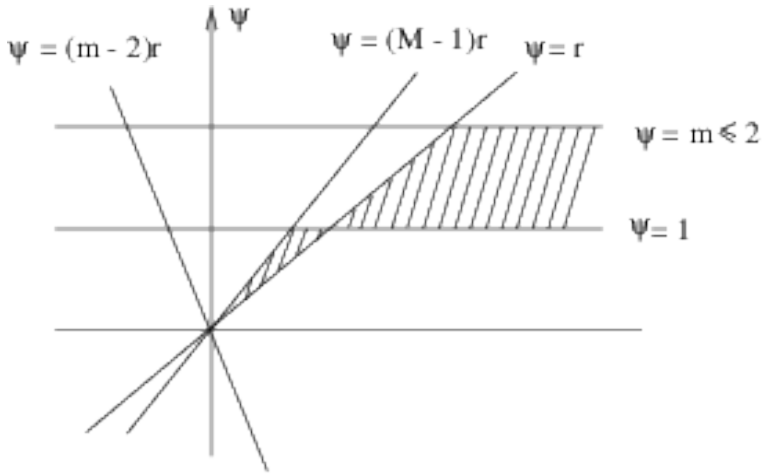


Figure 1.8: *Admissible region for graph of $\psi(r)$ for second order schemes.*

Normalized variable diagram for schemes of positive type

These conditions to impose on flux limiters to obtain schemes of positive type are easily transported to the normalized variable framework by means of the correspondence relations (1.73), (1.74). The normalized variable diagram corresponding to the flux limiter diagram of Fig. 1.8 is obtained as follows. The interval $0 \leq r < \infty$ corresponds to $0 \leq \tilde{\varphi} < 1$. The condition $\psi \leq m \leq 2$ corresponds to

$$f(\tilde{\varphi}) \leq \frac{m}{2} + (1 - \frac{m}{2})\tilde{\varphi} \,. \tag{1.91}$$

The condition $\psi \leq (M-1)r$ corresponds to

$$f(\tilde{\varphi}) \leq \frac{M+1}{2}\tilde{\varphi} \tag{1.92}$$

Condition (1.90) corresponds to

$$\begin{aligned} &\tfrac{1}{2}+\tfrac{1}{2}\tilde{\varphi} \geq f(\tilde{\varphi}) \geq \tfrac{3}{2}\tilde{\varphi}\,, \quad 0 \leq \tilde{\varphi} \leq \tfrac{1}{2}\,; \\ &\tfrac{1}{2}+\tfrac{1}{2}\tilde{\varphi} \leq f(\tilde{\varphi}) \leq \tfrac{3}{2}\tilde{\varphi}\,, \quad \tfrac{1}{2}\tilde{\varphi} \leq 1 \end{aligned} \tag{1.93}$$

The admissible region in the normalized variable diagram in the case $m = 2, M = 3$ is the interior of the dotted triangles in Fig. 1.5.

As noted before (cf. equation (1.67)), characteristics of second order schemes have to pass through $P = (1/2, 3/4)$. For third order accuracy the slope in P has to be equal to that of the $\kappa = 1/3$ scheme, i.e. $f'(1/2) = 5/6$. Finally, we want f to be continuous and f'' bounded on (0,1).

κ-scheme with limiting

A flux limited version of the κ-scheme with $f(\tilde{\varphi})$ having the desired properties may be obtained as follows. Outside (0,1), f is given by (1.71). On [0,1] one could fit a third degree polynomial through (0,0), (1/2, 3/4) and (1,1) with $f'(1/2)$ equal to the slope of the κ-scheme, i.e.

$$f'(1/2) = 1 - \kappa/2 \tag{1.94}$$

However, it turns out that this results in $f'(0) > 2$ for some values of κ, violating (1.90). This may also happen with a piecewise parabolic fit with a second degree polynomial through (0, 0) and (1/2, 3/4) satisfying (1.94). We therefore use a third degree polynomial on $[0, 1/2)$, in order to control $f'(0)$. A parabola on [1/2, 1] is fine for $\kappa \geq 0$, but gives an overshoot ($f(\tilde{\varphi}) > 1$) for $\kappa < 0$. In that case we use a third degree polynomial with $f'(1) = 0$. This results in:

$$f(\tilde{\varphi}) = \begin{cases} \tilde{\varphi}, & \tilde{\varphi} \notin (0,1) \\ 2\tilde{\varphi} + (\kappa-1)\tilde{\varphi}^2 - 2\kappa\tilde{\varphi}^3, & 0 \leq \tilde{\varphi} < 1/2 \\ 1 + \frac{1}{2}\kappa(\tilde{\varphi}-1) + (\kappa-1)(\tilde{\varphi}-1)^2, & 0 \leq \kappa \leq 1, \quad 1/2 \leq \tilde{\varphi} \leq 1 \\ 1 - (1+\kappa)(\tilde{\varphi}-1)^2 - 2\kappa(\tilde{\varphi}-1)^3, & -1 \leq \kappa < 0, \quad 1/2 \leq \tilde{\varphi} \leq 1 \end{cases}$$

The resulting characteristics are given in Fig. 1.5 for $\kappa = -1$ (second order upwind scheme), $\kappa = 0$ (Fromm scheme), $\kappa = 1/3$ (third order upwind biased scheme), $\kappa = 1/2$ (QUICK scheme) and $\kappa = 1$ (second order central scheme).

The corresponding limiter function is found to be, using (1.74) and (1.73),

$$\psi(r) = \begin{cases} 0, & r < 0 \\ \frac{2r}{(1+r)^2}\{1 + (1+\kappa)r\}, & 0 \leq r \leq 1 \\ \frac{1}{1+r}\{\kappa + (2-\kappa)r\}, & r > 1, \quad 0 \leq \kappa \leq 1 \\ \frac{2}{(1+r)^2}\{\kappa + (1-\kappa)r + r^2\}, & r > 1, \quad -1 \leq \kappa < 0 \end{cases}$$

According to experience somewhat better accuracy is obtained if the κ-scheme (i.e. the dashed lines in figure 1.5) is used as much as possible. We therefore choose the characteristic function $f(\tilde{\varphi})$ piecewise linear, similar to the SMART scheme proposed by Gaskell and Lau (1988), and we will call it PL. The PL characteristic function is shown in figure 1.5. This type of

characteristic function can be implemented very efficiently, as shown in Leonard (1995), in the following way. The median function $\mathrm{med}(a,b,c)$ takes three real numbers and returns the number that lies in between the other two or coincides with one of the other two. The median function is easily implemented as

$$\mathrm{med}(a,b,c) = \max[\min(a,b), \min\{\max(a,b), c\}]$$

From figure 1.5 it is clear that the graph of

$$f_1(\tilde{\varphi}) \equiv \mathrm{med}\{2\tilde{\varphi}, 1, \tilde{\varphi}\}$$

is given by $f_1(\tilde{\varphi}) = \tilde{\varphi}$ for $\tilde{\varphi} \notin (0,1)$ whereas otherwise the two upper sides of the triangle are followed. It follows that

$$f(\tilde{\varphi}) = \mathrm{med}(f, (\tilde{\varphi}), \frac{2-\kappa}{2}\tilde{\varphi} + \frac{1+\kappa}{4}, \tilde{\varphi})$$

gives the PL characteristic depicted in Fig. 1.5. Noting that $d + \mathrm{med}(a,b,c) = \mathrm{med}(a+d, b+d, c+d)$ and $d\,\mathrm{med}(a,b,c) = \mathrm{med}(da, db, dc)$ we can switch from normalized to original variables for efficiency:

$$\begin{aligned} f_2(\varphi_j) &= \mathrm{med}(2\varphi_j - \varphi_{j-1}, \varphi_j, \varphi_{j+1}) \\ \varphi_{j+1/2} &= \mathrm{med}(f_2(\varphi_j), \varphi^{\kappa}_{j+1/2}, \varphi_j) \\ \varphi^{\kappa}_{j+1/2} &= \frac{1}{2}(\varphi_j + \varphi_{j+1}) + \frac{1-\kappa}{4}(-\varphi_{j-1} + 2\varphi_j - \varphi_{j+1}) \end{aligned}$$

Nonlinear characteristics for the QUICK scheme ($\kappa = 1/2$) have been proposed in Leonard (1988) and Gaskell and Lau (1988). Some well-known flux limiter functions are:

Van Albada (1982):

$$\psi(r) = \frac{r^2 + r}{r^2 + 1}$$

Van Leer (1977b):

$$\psi(r) = \frac{r + |r|}{1 + |r|}$$

Roe (1986) ("superbee")

$$\psi(r) = \max\{0, \min(2r, 1), \min(r, 2)\}$$

Using (1.74) and (1.73) the corresponding characteristic functions are found to be, respectively:

$$\text{Van Albada:} \quad f(\tilde{\varphi}) = \tilde{\varphi} + \frac{\tilde{\varphi} - \tilde{\varphi}^2}{1 + (1 - 2\tilde{\varphi})^2}$$

$$\text{Van Leer:} \quad f(\tilde{\varphi}) = \begin{cases} 2\tilde{\varphi} - \tilde{\varphi}^2, & \tilde{\varphi} \in [0,1] \\ \tilde{\varphi}, & \tilde{\varphi} \notin [0,1] \end{cases}$$

$$\text{Roe :} \quad f(\tilde{\varphi}) = \begin{cases} 2\tilde{\varphi}, & 0 \le \tilde{\varphi} < \frac{1}{3} \\ \frac{1}{2}(1+\tilde{\varphi}), & \frac{1}{3} \le \tilde{\varphi} < \frac{1}{2} \\ \frac{3}{2}\tilde{\varphi}, & \frac{1}{2} \le \tilde{\varphi} < \frac{2}{3} \\ 1, & \frac{2}{3} \le \tilde{\varphi} \le 1 \\ \tilde{\varphi}, & \tilde{\varphi} \notin [0,1] \end{cases}$$

All of these characteristics are in the triangle of Fig. 1.5, and hence are wiggle-free and TVD. Also, they pass through (1/2, 3/4), but the second order accuracy of the "superbee" flux limiter is in doubt, because $f''(1/2)$ is unbounded; cf. the assumption preceding (1.66). The "superbee" scheme is known to suffer from "clipping", i.e. near a maximum the numerical solution shows a plateau slightly below the exact maximum, and analogously for minima. Because $f'(1/2)$ does not exist, the "superbee" scheme cannot be regarded as a nonlinear version of the κ-scheme. For the Van Albeda and Van Leer flux limiters we have $f'(1/2) = 1$, corresponding to $\kappa = 0$ (cf. (1.64)), so that these may be regarded as flux limited versions of the Fromm scheme.

Exercise 1.10.1 Let the grid be uniform. Determine $F^c_{j+1/2}$ by interpolating $\varphi_{j+1/2}$ to the highest possible order of accuracy between φ_{j-1}, φ_j and φ_{j+1}, and verify that this gives the $\kappa = 1/2$ scheme. Determine the order of the local truncation error in the approximation of $\mathrm{d}\varphi/\mathrm{d}x$ for $\kappa = 1/2$ and $\kappa = 1/3$. Next, determine $F^c_{1+1/2}$ by interpolation of $\varphi_{j+1/2}$ to highest order between $\varphi_{j-1}, ..., \varphi_{j+2}$. Determine the order of the resulting local truncation error in the approximation of $\mathrm{d}\varphi/\mathrm{d}x$. Do this also for (1.62), and compare.

Exercise 1.10.2 The following flux limiter is proposed by Roe (1986):

$$\psi(r) = \max\{0,\ \min(r, 1)\}$$

Determine the corresponding characteristic. Is the scheme of positive type? What is its order of accuracy?

1.11 'Convective' conclusions

It has been demonstrated that the discretization of convection, just a 'simple' first-order derivative, is very subtle. The major problem with the discretization is that, through its discretization error, it interferes with diffusion: sometimes this poses no serious harm, but sometimes it can also be devastating to the quality of the numerical solution. It is impossible to give general guidelines, yet I would like to do so.

The question to ask is:

"Is diffusion critical to the flow problem?"

Depending upon the answer, two situations arise:

Yes, diffusion is critical In this case it is necessary to resolve all boundary layers through grid refinement. Further, convection is preferably discretized with skew-symmetric methods.

No, diffusion is not critical In this case diffusion could be neglected (to obtain the Euler equations), but also there is no problem with (limited) increasing diffusion. Boundary layers do not have to be resolved, and upwind-biased discretization of convection (implicitly adding numerical diffusion) is applicable; explicit addition of diffusion is suitable as well. If necessary, to retain sharp discontinuities or to further suppress oscillations, limiters may be applied.

1.12 Appendix: Dirichlet–Neumann stability analysis

We will here[8] give the details of the stability analysis of (1.40) under mixed Dirichlet–Neumann conditions

$$\phi(0,t) = \text{given} \quad \text{and} \quad \frac{\partial \phi}{\partial x}(1,t) = \text{ given.}$$

(1.40) combined with these boundary conditions can be written in following matrix notation

$$2I\,\phi^{(n+1)} = (2I - A)\,\phi^{(n)} \tag{1.95}$$

with

$$A = \begin{pmatrix} 2d & \eta - d & & & \\ -\eta - d & 2d & \eta - d & & \\ & & \ddots & & \\ & & -\eta - d & 2d & \eta - d \\ & & & -2d & 2d \end{pmatrix} \begin{array}{l} \longleftarrow \text{ here the Dirichlet condition} \\ \quad\ \text{has been substituted} \\ \\ \\ \\ \longleftarrow \text{ the Neumann condition} \end{array} \tag{1.96}$$

We need two lemmas. The first lemma introduces a similarity transformation with which A can be transformed into either a symmetric matrix or the sum of a constant diagonal matrix and a skew-symmetric matrix. The second lemma gives an estimate for the eigenvalues of this type of matrices. We remark that Lemma 1 implies that A is non-defect.

Lemma 1 Let A be a tridiagonal matrix with

$$\begin{gathered} a_{ii} = \alpha_i \quad (i = 1, \ldots, k); \\ a_{i+1,i} = \gamma_{i+1}, \quad a_{i,i+1} = \beta_{i+1}, \quad \gamma_i \neq 0, \quad \beta_i \neq 0 \quad (i = 1, \ldots, k-1). \end{gathered}$$

Let D be a diagonal matrix with

$$d_{11} = 1, \quad d_{ii} = d_{i-1,i-1}\sqrt{\left|\frac{\gamma_i}{\beta_i}\right|} \quad (i = 2, \ldots, k).$$

Let $B = D^{-1}AD$. Then B is tridiagonal with

$$\begin{gathered} b_{ii} = \alpha_i \quad (i = 1, \ldots, k); \\ \left.\begin{aligned} b_{i+1,i} &= \text{sgn}\ (\gamma_{i+1})\ \sqrt{|\beta_{i+1}\gamma_{i+1}|} \\ b_{i,i+1} &= \text{sgn}\ (\beta_{i+1})\ \sqrt{|\beta_{i+1}\gamma_{i+1}|} \end{aligned}\right\} \quad (i = 1, \ldots, k-1). \end{gathered}$$

Proof Is left to the reader. □

Lemma 2 Let B be a tridiagonal $k \times k$ matrix with

$$\begin{aligned} b_{ii} &= d \quad (i = 1, \ldots, k); \\ |b_{i+1,i}| &= |b_{i,i+1}| = b \quad (i = 1, \ldots, k-2); \\ |b_{k,k-1}| &= |b_{k-1,k}| = \gamma \text{ where } \gamma \leq b\sqrt{2}. \end{aligned}$$

Then each eigenvalue μ of B satisfies $|\mu - d| \leq 2b$.

Proof Let D be a diagonal matrix with $d_{ii} = 1 \ \ (i = 1, \ldots, k-1)$, and $d_{kk} = b/\gamma$. Let $C = D^{-1}BD$. We have $c_{ij} = b_{ij}, (i,j) \neq (k,k-1), (k-1,k)$ and $c_{k,k-1} = \frac{\gamma}{b} b_{k,k-1}$ en $c_{k-1,k} = \frac{b}{\gamma} b_{k-1,k}$. On C Gerschgorin's theorem can be applied from which the lemma follows immediately. □

[8] This section is taken from P. Wesseling and P. Wilders (1984) Numerieke Stromingsleer A, Lecture Notes a110A (in Dutch), TU Delft.

We can now proceed to estimating the eigenvalues of the matrix A from (1.96). When $\eta^2 \neq d^2$ the transformation from Lemma 1 is applicable. The resulting matrix B becomes

$$\begin{aligned}
b_{ii} &= 2d \quad (i = 1, \ldots, k); \\
b_{i,i+1} &= \text{sgn}\,(\eta - d)\,\sqrt{|d^2 - \eta^2|}\,, \quad (i = 1, \ldots, k-2); \\
b_{i+1,i} &= \text{sgn}\,(-\eta - d)\,\sqrt{|d^2 - \eta^2|}\,, \quad (i = 1, \ldots, k-2); \\
b_{k-1,k} &= \text{sgn}\,(\eta - d)\,\sqrt{2d\,|d - \eta|}\,; \\
b_{k,k-1} &= \text{sgn}\,(-2d)\,\sqrt{2d\,|d - \eta|}.
\end{aligned}$$

Since $\eta \geq 0$ we have

$$\sqrt{2d\,|d-\eta|} \leq \sqrt{2(d+\eta)\,|d-\eta|} = \sqrt{2}\,\sqrt{|d^2 - \eta^2|},$$

with which the matrix B satisfies the conditions from Lemma 2. We remark that the matrices A and B from Lemma 1 possess the same eigenvalues

$$\mu = \mu_R + \mathrm{i}\,\mu_I.$$

– For $d^2 > \eta^2$ the matrix B is symmetrical. The eigenvalues μ of B (and of A) are real and, according to Lemma 2, satisfy

$$2d - 2\sqrt{d^2 - \eta^2} \leq \mu_R \leq 2d + 2\sqrt{d^2 - \eta^2}, \quad \mu_I = 0.$$

– For $d^2 < \eta^2$ we have $B = 2dI + \bar{B}$, where $\bar{B}$ is skew symmetric. The eigenvalues of $\bar{B}$ are purely imaginary. Lemma 2 is applicable to $\bar{B}$, which yields

$$\mu_R = 2d, \quad 0 \leq \mu_I \leq 2\sqrt{\eta^2 - d^2}. \tag{1.97}$$

Absolute stability requires all eigenvalues λ of the iteration matrix $(I - \frac{1}{2}A)$ from (1.95) to satisfy $|\lambda| \leq 1$. I.e. the eigenvalues of A (and B) have to satisfy

$$|1 - \tfrac{1}{2}\mu| \leq 1 \quad \Leftrightarrow \quad \mu_R^2 + \mu_I^2 - 4\mu_R \leq 0. \tag{1.98}$$

– For $d^2 > \eta^2$ we have $\mu_I = 0$, after which we only have to show that $0 \leq \mu_R \leq 4$. From (1.97) it follows simply that

$$d + \sqrt{d^2 - \eta^2} \leq 2$$

is a sufficient condition to satisfy (1.98).

– For $d^2 < \eta^2$ we have $\mu_R = 2d$. The estimate (1.97) now yields that

$$\eta^2 \leq 2d$$

is a sufficient condition for (1.98) to hold.

Herewith the statement in (1.48) has been proven. Moreover it is clear, from a comparison with (1.43), that the Dirichlet-Neumann conditions yield additional stability in comparison with periodic boundary conditions.

1.13 References

G. van Albada, B. van Leer and W. Roberts (1982) A comparative study of computational methods in cosmic gas dynamics. *Astron. Astrophys.* 108, 76–84.

D.N. de G. Allen and R.V. Southwell (1955) Relaxation methods applied to determine the motion in two dimensions of a viscous fluid past a fixed cylinder. *Quart. J. Mech. Appl. Math.* 8, 129

W. Anderson, J. Thomas and B. van Leer (1985) A comparison of finite volume flux vector splittings for the Euler equations. AIAA Paper 85-0122.

Fromm, J. (1968) A method for reducing dispersion in convective difference schemes. *J. Comput. Phys.* 3, 176–189.

P. Gaskell and K. Lau (1988) Curvature-compensated convective transport: SMART, a new boundedness-preserving transport algorithm. *Int. J. Numer. Meth. Fluids* 8, 617–641.

P.M. Gresho and R.L. Lee (1981) Don't suppress the wiggles - they are telling you something. *Comput. Fluids* 9, 223–253.

A.C. Hindmarsch, P.M. Gresho and D.F. Griffiths (1984) The stability of explicit Euler time-integration for certain finite difference approximations of the multi-dimensional advection-diffusion equation. *Int. J. Num. Meth. Fluids* 4, 853–897.

C. Hirsch (1990) *Numerical Computation of Internal and External Flows, Volume 2.* John Wiley.

A. Jameson, W. Schmidt and E. Turkel (1981) Numerical solutions of the Euler equations by finite-volume methods using Runge–Kutta time-stepping schemes. *AIAA paper 81-1259.*

B. van Leer (1974) Towards the ultimate conservative difference scheme. II. Monitonicity and conservation combined in a second order scheme. *J. Comput. Phys.* 14, 361–370.

B. van Leer (1977a) Towards the ultimate conservative difference scheme III. Upstream-centered finite-difference schemes for ideal compressible flow. *J. Comput. Phys.* 23, 263–275.

B. van Leer (1977b) Towards the ultimate conservative difference scheme IV. A new approach to numerical convection. *J. Comput. Phys.* 23, 276–299.

B. van Leer (1985) Upwind-difference methods for aerodynamic problems governed by the Euler equations. *Lectures in Appl. Math.* 22, 327–336.

B.P. Leonard (1979a) A stable and accurate convective modelling procedure based on quadratic upstream interpolation. *Comput. Meth. Appl. Mech. Eng.* 19, 59–98.

B.P. Leonard (1979b) A survey of finite differences of opinion on numerical muddling of the incomprehensible defective confusion equation. *Finite Element Methods for Convection Dominated Flows*, Vol. 34, 1–10, ASME New York.

B.P. Leonard (1988) Simple high-accuracy resolution program for convective modelling of discontinuities. *Int. J. Num. Meth. Fluids* 8, 1291–1318.

B.P. Leonard and J.E. Drummond (1995) Why you should not use 'hybrid', 'power-law' or related exponential schemes for convective modelling - there are much better alternatives. *Int. J. Num. Meth. Fluids* 20, 421–442.

T.A. Manteuffel and A.B. White Jr. (1986) The numerical solution of second-order boundary value problems on nonuniform meshes. *Math. Comput.* 47, 511–535.

R.D. Richtmyer and K.W. Morton (1967) *Difference Methods for Initial Value Problems.* John Wiley.

P. Roe (1986) Characteristic-based schemes for the Euler equations. In M. Van Dyke *et al.* (Ed.), *Annual Review of Fluid Mechanics Vol.18*, 337–365. Annual Reviews Inc.

D. Spalding (1972) A novel finite-difference formulation for differential expressions involving both first and second derivatives. *Int. J. Num. Meth. Eng.* 4, 551–559.

P. Sweby (1984) High resolution schemes using flux-limiters for hyperbolic conservation laws. *SIAM J. Num. Anal.* 21, 995–1011.

A.E.P. Veldman and E.F.F. Botta (1993) Grid quality - an interplay of grid and solver. In: M.J. Baines and K.W Morton (eds.) *Numerical Methods for Fluid Dynamics 4.* Oxford University Press, 329–335.

A.E.P. Veldman and K. Rinzema (1991) Playing with nonuniform grids, *J. Eng. Math.* 26, 119–130.

R.W.C.P. Verstappen and A.E.P. Veldman (1997) Direct numerical simulation of turbulence at lower costs. *J. Eng. Math.* 32, 143–159.

R.W.C.P. Verstappen and A.E.P. Veldman (1998) Spectro-consistent discretization: a challenge to RANS and LES. *J. Eng. Math.* 34, 163–179.

R.W.C.P. Verstappen and A.E.P. Veldman (2003) Symmetry-preserving discretization for turbulent flow. *J. Comput. Phys.* 187, 343–368.

R. Warming and R. Beam (1976) Upwind second-order difference schemes and applications in aerodynamic flows. *AIAA J.* 14, 1241–1249.

P. Wesseling (1973) On the construction of accurate difference schemes for hyperbolic partial differential equations. *J. Eng. Math.* 7, 1–31.

P. Wesseling (2001) *Principles of Computational Fluid Dynamics*. Springer Verlag.

F.W. Wubs and J. Thies (2011) A robust two-level incomplete factorization for (Navier–)Stokes saddle point matrices. *SIAM J. Matrix Anal. Appl.* 32, 1475–1499.

Chapter 2

Incompressible Navier–Stokes equations

2.1 The equations for fluid flow

The equations of fluid flow are governed by conservation laws for mass, momentum and energy, completed by a number of thermodynamic relations. The conservation law of mass in a volume Ω reads

$$\int_\Omega \frac{\partial \rho}{\partial t}\,\mathrm{d}\Omega + \int_\Gamma \rho\,\boldsymbol{u}\cdot\boldsymbol{n}\,\mathrm{d}\Gamma = 0, \tag{2.1}$$

where ρ denotes the density of the fluid. Conservation of momentum can be written as

$$\int_\Omega \frac{\partial \rho\boldsymbol{u}}{\partial t}\,\mathrm{d}\Omega + \int_\Gamma \rho\boldsymbol{u}\,\boldsymbol{u}\cdot\boldsymbol{n}\,\mathrm{d}\Gamma = \int_\Gamma \boldsymbol{\sigma}\cdot\boldsymbol{n}\,\mathrm{d}\Gamma, \tag{2.2}$$

where $\boldsymbol{\sigma} = -p\boldsymbol{I} + \boldsymbol{\tau}$ is the internal stress tensor, with p the pressure. The viscous stress tensor $\boldsymbol{\tau}$ is defined through

$$\boldsymbol{\tau} = \mu[\nabla\boldsymbol{u} + (\nabla\boldsymbol{u})^* - \tfrac{2}{3}(\nabla\cdot\boldsymbol{u})\,\boldsymbol{I}],$$

with μ the dynamic viscosity of the fluid. The conservation law of energy $E = e + \frac{1}{2}\boldsymbol{u}\cdot\boldsymbol{u}$ reads

$$\int_\Omega \frac{\partial \rho E}{\partial t}\,\mathrm{d}\Omega + \int_\Gamma \rho E\,\boldsymbol{u}\cdot\boldsymbol{n}\,\mathrm{d}\Gamma = \int_\Gamma (\boldsymbol{\sigma}\cdot\boldsymbol{u})\cdot\boldsymbol{n}\,\mathrm{d}\Gamma + \int_\Gamma k\frac{\partial T}{\partial n}\,\mathrm{d}\Gamma,$$

where heat transport due to temperature (T) differences has been modelled by Fourier's law, with k being the thermal conductivity coefficient. Finally, two thermodynamic relations are applicable

$$p = \rho R T \quad \text{and} \quad e = c_v T,$$

where $R = c_p - c_v$ is the universal gas constant, and c_p and c_v are the specific heat coefficients.

Several dimensionless parameters are associated with fluid flow. For viscous flow the most important one is the Reynolds number, named after Osborne Reynolds (1842-1912). It is defined by $\mathrm{Re} = \rho U L/\mu$, where U is a characteristic velocity of the flow and L a characteristic length scale. Roughly, this number indicates the relevance of viscous effects in the flow: the larger Re, the less influence viscosity has on the flow. In this vain, the Reynolds number forms a useful indicator for the onset of turbulence (see Chapter 3).

Differential form By applying Gauss' theorem, the above conservation laws can be transformed into differential equations. In particular, conservation of mass (2.1) and momentum (2.2) can be reformulated as

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{u}) = 0 \tag{2.3}$$

and

$$\frac{\partial \rho \boldsymbol{u}}{\partial t} + \nabla \cdot (\rho \boldsymbol{u}\, \boldsymbol{u}) = -\nabla p + \nabla \cdot \boldsymbol{\tau}. \tag{2.4}$$

By combining both equations, conservation of momentum can also be written in the non-conservative form

$$\rho \frac{\partial \boldsymbol{u}}{\partial t} + \rho \boldsymbol{u} \cdot \nabla \boldsymbol{u} = -\nabla p + \nabla \cdot \boldsymbol{\tau}. \tag{2.4$'$}$$

Incompressible flow

In these lecture notes we will focus on incompressible flow. In this case, effectively the energy equation is replaced by the condition of incompressibility

$$\frac{D\rho}{Dt} \equiv \frac{\partial \rho}{\partial t} + \boldsymbol{u} \cdot \nabla \rho = 0,$$

i.e. the density of a given particle remains constant during its motion (note that in principle the density does not have to be the same everywhere in the flow). This condition can be combined with the conservation of mass in differential form (2.3) to conclude that the velocity field is divergence free[1]

$$\operatorname{div} \boldsymbol{u} = 0. \tag{2.5}$$

The stress tensor $\boldsymbol{\tau}$ now can be simplified, and when moreover ρ and μ are assumed constant, the momentum equation (2.4$'$) becomes

$$\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u} \cdot \operatorname{grad}) \boldsymbol{u} = -\frac{1}{\rho} \operatorname{grad} p + \nu \operatorname{div} \operatorname{grad} \boldsymbol{u}, \tag{2.6}$$

where $\nu = \mu/\rho$ is the kinematic viscosity. It is remarked that under the incompressibility constraint (2.5), the convective term can be written as $(\boldsymbol{u} \cdot \operatorname{grad}) \boldsymbol{u} = \operatorname{div}(\boldsymbol{u}\, \boldsymbol{u})$. Now the momentum equation can again be denoted in conservation form

$$\frac{\partial \boldsymbol{u}}{\partial t} + \operatorname{div}(\boldsymbol{u}\boldsymbol{u}) = -\frac{1}{\rho} \operatorname{grad} p + \nu \operatorname{div} \operatorname{grad} \boldsymbol{u}. \tag{2.6$'$}$$

When an internal flow is simulated in a domain Ω with a solid boundary Γ only a boundary condition for the velocity $\boldsymbol{u}$ can be formulated. Usually this will be

$$\boldsymbol{u} = 0 \quad \text{on } \Gamma. \tag{2.7}$$

This boundary condition describes two physical effects. Firstly, the impenetrability of the wall requires the normal velocity to vanish. Secondly, the (viscous) shear stress requires the tangential velocity component to vanish (no-slip condition).

[1] From now on we will use the notation 'div' and 'grad' instead of the '∇'-notation.

Acoustical degeneration

Heuristically there does not seem to be an equation for the pressure, unlike in the compressible case. It is natural to consider the momentum equation as an equation for the velocity. In the compressible case the continuity equation is an equation for the density, whereas the pressure then follows through an equation of state. In the incompressible case the density does not appear in the continuity equation (2.5), and neither does the pressure. But of course (2.5)+(2.6) together determine velocity and pressure.

To get a feeling for what is going on, we will first consider the one-dimensional isentropic Euler equations

$$\begin{cases} \dfrac{\partial \rho}{\partial t} + \dfrac{\partial}{\partial x}(\rho u) = 0, \\ \dfrac{\partial u}{\partial t} + u\dfrac{\partial u}{\partial x} + \dfrac{1}{\rho}\dfrac{\partial p}{\partial x} = 0, \end{cases} \tag{2.8}$$

where pressure and density are related through

$$p = p(\rho) \quad \text{with} \quad \frac{\mathrm{d}p}{\mathrm{d}\rho} = c^2 > 0, \tag{2.9}$$

in which c is the velocity of sound. Substitution of the p-ρ relation (2.9) in (2.8) yields

$$\begin{cases} \dfrac{\partial \rho}{\partial t} + \rho\dfrac{\partial u}{\partial x} + u\dfrac{\partial \rho}{\partial x} = 0, \\ \dfrac{\partial u}{\partial t} + u\dfrac{\partial u}{\partial x} + \dfrac{c^2}{\rho}\dfrac{\partial \rho}{\partial x} = 0. \end{cases} \tag{2.10}$$

This system of equations possesses the characteristic directions (found by diagonalizing the coefficient matrix; see any book on partial differential equations)

$$\frac{\mathrm{d}x}{\mathrm{d}t} = u + c \text{ and } u - c.$$

Both directions are real, hence the system is hyperbolic. In the incompressible limit we have $c \to \infty$, therefore the two characteristic directions $\mathrm{d}x/\mathrm{d}t$ approach $\pm\infty$, and hence coincide in the limit with the x-axis and with each other.

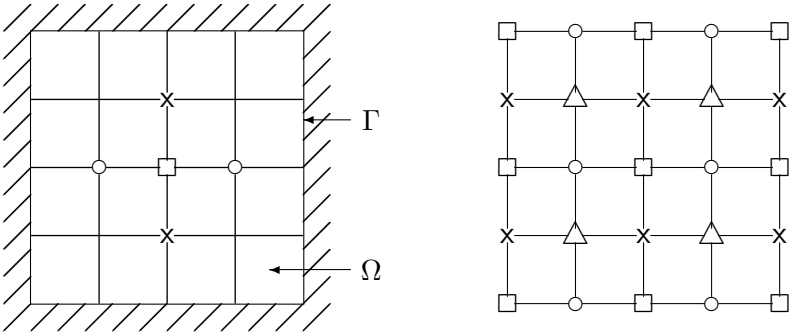
The theory on partial differential equations now tells us that the system is degenerating. The acoustical part of the equations, the combination of div $\boldsymbol{u}$ and grad p, is responsible for this behaviour. Disturbances in the pressure are instantaneously noted everywhere in the domain. This infinite propagation speed will force us to treat the acoustical part of the Navier–Stokes equations in an implicit manner, as we will see in the sequel.

2.2 Choice of the computational grid

We will start the numerical treatment with a discussion of the spatial discretization. For convenience only rectangular domains Ω will be considered, covered by a uniform grid Ω. The grid cells are positioned such that the cell faces coincide with the boundary Γ of Ω. It seems obvious to define the unknown variables $\boldsymbol{u} = (u, v)$ and p in the corners of the grid cells. However, there are some snakes in the grass. This is not only the case in finite-difference or finite-volume discretizations, but also in a finite-element approach.

A: u, v and p in corner points

When u, v and p are defined in the corner points of the grid cells, it is obvious to apply the momentum equations in these corner points too. For second-order accuracy, the discrete version of grad p, which we shall denote by $\boldsymbol{G}p$, should be computed with central differences. For the grid point indicated with $\square$ this implies that values of p are used from grid points indicated by $\circ$ (for $\partial p/\partial x$) and $\times$ (for $\partial p/\partial y$). In this discretization it is possible to construct a $\tilde{p} \neq$ Constant for which $\boldsymbol{G}\tilde{p} = 0$. To each solution p this $\tilde{p}$ can be added without any consequences for the equations of motion. Moreover there is the danger that during the computations one encounters (4-fold) singular matrices (i.e. $\dim \mathrm{Ker}(\boldsymbol{G}) = 4$), which reflect this non-uniqueness of the solution.



In the above figures, the grid points have been divided into 4 groups: $\square$, $\circ$, $\times$ and $\triangle$. Choose $\tilde{p}$ such that in each of the groups $\tilde{p}$ is constant, but the 4 constants are allowed to be arbitrary. Then in a discrete sense $\boldsymbol{G}\tilde{p} = 0$, while $\tilde{p}$ shows oscillating behaviour.

Again we encounter an example of odd/even decoupling, caused by discretizing a first-order derivative over two grid cells. When p itself, and not just its gradient, is relevant to the problem (e.g. one would like to know forces exerted on the solid walls), then some kind of averaging has to be applied to remove the non-physical oscillations. This is not very elegant.

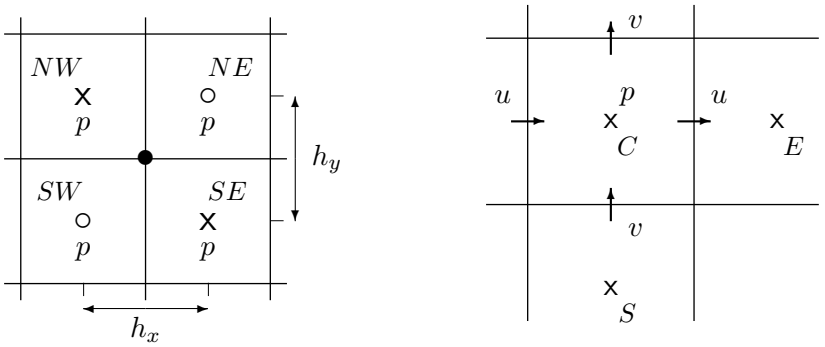
In spite of the above drawback, this spatial positioning is often used, especially for compressible flow. Its advantage is that all flow variables (velocity components, pressure, density, temperature, etc.) are defined at the same location. Thus relations between these variables, for instance the thermodynamic relations, are easily implemented without complicated averaging procedures. However, to overcome the above decoupling of the pressure a price has to be paid, for instance some form of artificial 'pressure diffusion' (e.g. Rhie and Chow 1983). An alternative is to filter out the pressure component that lies in $\mathrm{Ker}(\mathbf{G})$.

B: u and v in corner points, p in cell centres

An alternative is to define p in the centres of the grid cells. In the momentum equation applied in the point $\bullet$, second-order discretization of grad p leads to

$$\begin{aligned}
\frac{\partial p}{\partial x} &\approx \frac{1}{2}\left\{\frac{p_{NE}-p_{NW}}{h_x}+\frac{p_{SE}-p_{SW}}{h_x}\right\} = \frac{1}{2h_x}\left\{p_{NE}-p_{SW}+p_{SE}-p_{NW}\right\}, \\
\frac{\partial p}{\partial y} &\approx \frac{1}{2}\left\{\frac{p_{NE}-p_{SE}}{h_y}+\frac{p_{NW}-p_{SW}}{h_y}\right\} = \frac{1}{2h_y}\left\{p_{NE}-p_{SW}-p_{SE}+p_{NW}\right\}. \qquad (2.11)
\end{aligned}$$

(2.11) implies that a discrete pressure $\tilde{p}$, of which the values in the points indicated with $\times$ are equal and likewise in the points indicated with $\circ$, will lead to $\boldsymbol{G}\tilde{p} = 0$, i.e. $\dim \mathrm{Ker}(\boldsymbol{G}) = 2$. The same problems as described above are present, this time with a decoupling according to a checkerboard mode (red-black decoupling).

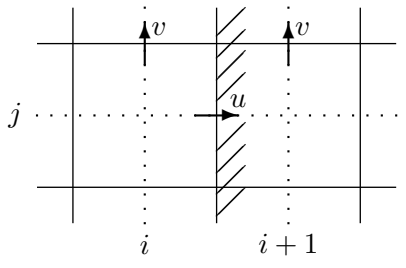


Alternative positionings B (left) and C (right) for the unknowns of Navier–Stokes.

C: u and v in middle of cell faces, p in cell centres

Yet another alternative is to define the pressure in the centre of a cell, u in the middle of the vertical cell faces, and v in the middle of the horizontal faces. This arrangement of velocities and pressure is called a *staggered grid*. The x-momentum equation is applied in the point where u is defined; similar for the y-momentum equation. For alternatives A and B a $p \neq$ Constant can be found for which $\boldsymbol{G}p = 0$. Here this is no longer possible, since $\partial p/\partial x = 0$ applied in the middle of the right-hand cell face (where the x-momentum equation is applied) yields $p_E = p_C$, whereas $\partial p/\partial y = 0$ in the middle of the lower cell face yields $p_S = p_C$. In this way all pressure points are coupled to each other, and the only mode in the kernel of the discrete gradient is the constant function. Note that the continuous gradient operator has the same kernel (for incompressible flow the pressure is determined up to a constant), hence we cannot improve the situation further.

A minor disadvantage of alternative C is that boundary conditions for the velocity do not directly correspond with the velocities in 'their' grid points. On a vertical wall, for example, the point where the horizontal velocity is defined is on the boundary (hence no problem here), but the point where the vertical velocity is defined is half a mesh size away from the boundary.



A second-order discretization of the boundary condition for the vertical velocity requires the use of a mirror point. The discrete boundary condition $(u, v) = (0, 0)$ in the indicated situation becomes

$$\begin{aligned} (u)_{i+\frac{1}{2},j} &= 0, \\ (v)_{i,j+\frac{1}{2}} &= -(v)_{i+1,j+\frac{1}{2}}. \end{aligned}$$

Alternative C is used very often, for example in the methods derived from the Marker-And-Cell (MAC) method developed in the mid-sixties (Harlow and Welsh 1965). This method was developed to describe free-surface flow. It contained two new ideas: one is the staggering of the discrete locations as just described, the other is the way in which markers are used to

keep track of the liquid. The latter idea gave the method its name. Another early application is in meteorology (e.g. Arakawa 1966), where the three above schemes are known as Arakawa A, B and C, respectively.

2.3 Discretization - explicit

In the sequel the computational grid from the MAC method (alternative C) will be used. However, this does not imply that both other alternatives do not have reason for existence (for example on non-orthogonal grids). The grids remain uniform for ease of presentation.

First the equations of motion (2.5)+(2.6) will be written in a more schematic form

$$\operatorname{div} \boldsymbol{u} = 0, \tag{2.12}$$

$$\frac{\partial \boldsymbol{u}}{\partial t} + \operatorname{grad} p = \boldsymbol{R}, \tag{2.13}$$

where p/ρ is replaced by p (i.e. ρ is normalized at 1). Further

$$\boldsymbol{R} = -(\boldsymbol{u} \cdot \operatorname{grad})\, \boldsymbol{u} + \nu \operatorname{div} \operatorname{grad} \boldsymbol{u} \tag{2.14}$$

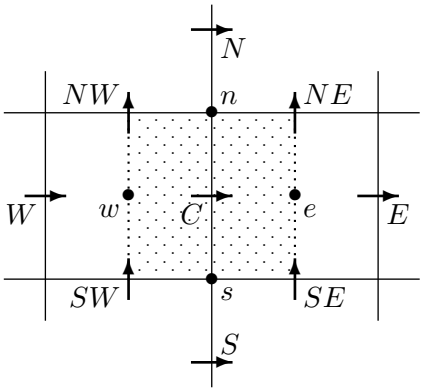
contains all convective and diffusive forces; if desired, also a body force can be included.

In this section, again for simplicity, an explicit time-integration method with two time levels is applied:

$$\operatorname{div} \boldsymbol{u}^{(n+1)} = 0, \tag{2.15}$$

$$\frac{\boldsymbol{u}^{(n+1)} - \boldsymbol{u}^{(n)}}{\delta t} + \operatorname{grad} p^{(n+1)} = \boldsymbol{R}^{(n)}. \tag{2.16}$$

Equation (2.15) and $\operatorname{grad} p$ in (2.16) have received an index $n+1$ to indicate that the $\boldsymbol{u}^{(n+1)}$ that follows from (2.16) is divergence free. In fact, the constraint (2.15) determines the pressure $p^{(n+1)}$: p is the Lagrange multiplier corresponding with this constraint.



For the discretization in space the following schemes are applied (for notation see accompanying figure):

Continuity equation

The continuity equation (2.15) is applied in cell centres. In a natural way it can be discretized centrally. The eastern cell with centre e gives a discrete finite-volume conservation law

$$u_E^{(n+1)}\, h_y + v_{NE}^{(n+1)}\, h_x - u_C^{(n+1)}\, h_y - v_{SE}^{(n+1)}\, h_x = 0. \tag{2.17}$$

In finite-difference notation it reads

$$\frac{u_E^{(n+1)} - u_C^{(n+1)}}{h_x} + \frac{v_{NE}^{(n+1)} - v_{SE}^{(n+1)}}{h_y} = 0. \tag{2.17$'$}$$

Observe that the number of continuity equations equals the number of grid cells. Also the number of pressure unknowns equals the number of cells. Hence, there are as many discrete continuity equations as unknown pressure values! Note that this equality does not hold for positioning methods A and B.

Momentum equation: convection

The x-momentum equation is applied in the midpoints of the vertical cell faces. In the point C we obtain (in finite-difference notation)

$$\frac{u_C^{(n+1)} - u_C^{(n)}}{\delta t} + \frac{p_e^{(n+1)} - p_w^{(n+1)}}{h_x} = R_C^{(n)}, \tag{2.18}$$

where the pressure gradient has been discretized centrally. The term R_C contains derivatives for the convective and diffusive terms; the latter are also discretized centrally. For the discretization of the convective terms several options exist, which will be shortly discussed next.

Written out, the convective term in the x-momentum equation (2.6) reads

$$u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y}, \tag{2.19}$$

which, since the velocity is divergence free, is identical to the conservation form

$$\frac{\partial}{\partial x}(u^2) + \frac{\partial}{\partial y}(uv); \tag{2.19$'$}$$

compare (2.6$'$). Both upwind and central discretization can be used. Their pro's and con's have been treated in Section 1.4.

In the discretization of (2.19) and (2.19$'$) some averages have to be computed, since u and v are not defined in the same point. For example the v appearing in $v\,\partial u/\partial x$ in (2.19) is computed as

$$v_C = \tfrac{1}{4}(v_{NE} + v_{NW} + v_{SW} + v_{SE}).$$

Some more remarks on the discretization in conservation form (2.19$'$) are in order. The control volume is chosen as indicated in the figure above. The total convective flux through the faces is given by

$$u_e\,u_e\,h_y + v_n\,u_n\,h_x - u_w\,u_w\,h_y - v_s\,u_s\,h_x. \tag{2.20}$$

Note that we have written $u_e\,u_e$ instead of u_e^2, since one of the factors is regarded as the transport velocity, while the other is the convected quantity, i.e. x-momentum. For the latter we choose

$$u_e = \tfrac{1}{2}(u_E + u_C), \quad u_n = \tfrac{1}{2}(u_N + u_C), \text{ etc.} \tag{2.21}$$

Substitution in (2.20) yields a convective flux

$$\tfrac{1}{2}[u_e(u_E + u_C) - u_w(u_W + u_C)]h_y + \tfrac{1}{2}[v_n(u_N + u_C) - v_s(u_S + u_C)]h_x. \tag{2.22}$$

The coefficient of u_C is given by $\frac{1}{2}[(u_e - u_w)h_y + (v_n - v_s)h_x]$. By once more substituting (2.21), and using the discrete continuity equations (2.17) in the eastern and western cells, this coefficient is found to vanish. Herewith the discrete version of the convective terms in finite-volume formulation becomes

$$\tfrac{1}{2}h_y(u_e\,u_E - u_w\,u_W) + \tfrac{1}{2}h_x(v_n\,u_N - v_s\,u_S). \tag{2.23}$$

Pressure gradient

Analytically, there exists a relation between the gradient operator and the divergence operator. Hereto, consider one-times differentiable functions defined on a domain Ω, for which at the boundary Γ of the domain the values of function and derivative vanish (in abstract mathematical terms, consider the function space $H_0^1(\Omega)$ of differentiable functions with compact support). Then Gauss' theorem gives

$$0 = \int_\Gamma \phi\, \boldsymbol{u} \cdot \boldsymbol{n}\, \mathrm{d}\Gamma = \int_\Omega \mathrm{div}\,(\phi\, \boldsymbol{u})\, \mathrm{d}\Omega = \int_\Omega (\phi\ \mathrm{div}\, \boldsymbol{u} + \mathrm{grad}\, \phi \cdot \boldsymbol{u})\, \mathrm{d}\Omega.$$

It follows that for functions which vanish near the boundary

$$\int_\Omega \mathrm{grad}\, \phi \cdot \boldsymbol{u}\, \mathrm{d}\Omega = -\int_\Omega \phi\ \mathrm{div}\, \boldsymbol{u}\, \mathrm{d}\Omega. \tag{2.24}$$

Recognizing that the integrals represent inner products, this relation can also be denoted as $\mathrm{grad}^* = -\mathrm{div}$. In words, relation (2.24) implies that, apart from a minus sign, the gradient operator and the divergence operator are each others adjoint[2]. With the current choice of discretization, this analytical relation also holds in the discrete approximation, as shown next. We will see below that this has favourable consequences.

In the discrete version, the inner-product integrals in (2.24) have to be considered in a discrete sense. Hence, firstly the products of the functions appearing in the respective integrands should make sense[3]. And, indeed with $\phi = p$, the discrete gradient $\boldsymbol{G}p$ and the discrete velocity vector $\boldsymbol{u}$ are defined at the same locations in the grid (cell edges). For p and the discrete divergence $\boldsymbol{D}\boldsymbol{u}$ the same applies (cell centers), hence their respective products are formed in a natural way. Further, it is not difficult to verify that the discrete matrices $\boldsymbol{G}$ and $-\boldsymbol{D}$, where apart from mesh-size scaling each row consists of a '1' and a '-1', are each others transpose, i.e. $\boldsymbol{G}^* = -\boldsymbol{D}$.

Symmetry-preserving discretization

When the variables with lower case subscripts in (2.23) are considered to be part of the coefficient matrix, whereas the upper case subscripts belong to the unknowns, a skew-symmetric contribution to the coefficients is recognized. With this definition, the discrete Navier–Stokes equation can be written as

$$\boldsymbol{\Omega}\frac{\mathrm{d}\boldsymbol{u}}{\mathrm{d}t} + \boldsymbol{C}(\boldsymbol{u})\,\boldsymbol{u} + \boldsymbol{V}\boldsymbol{u} + \boldsymbol{G}p = \boldsymbol{0}, \qquad \boldsymbol{D}\boldsymbol{u} = 0, \tag{2.25}$$

[2]Let $\mathcal{V}$ and $\mathcal{W}$ be two function spaces and let the operator $\mathcal{A} : \mathcal{V} \to \mathcal{W}$. Then the adjoint operator $\mathcal{A}^* : \mathcal{W} \to \mathcal{V}$ is defined by $(\mathcal{A}v, w)_\mathcal{W} = (v, \mathcal{A}^* w)_\mathcal{V}$ for all $v \in \mathcal{V}$ and $w \in \mathcal{W}$, where $\langle\cdot,\cdot\rangle$. denotes the inner product in the respective function spaces.

[3]In a more mathematical notation, define two discrete function spaces:

i) $\mathcal{V}_{\mathrm{div}}$: the space of scalar-valued functions defined in cell centers;

ii) $\mathcal{V}_{\mathrm{grad}}$: the space of vector-valued functions defined at cell faces.

As an example, we have $p \in \mathcal{V}_{\mathrm{div}}$ and $\boldsymbol{u} \in \mathcal{V}_{\mathrm{grad}}$. Next note that $\boldsymbol{G} : \mathcal{V}_{\mathrm{div}} \to \mathcal{V}_{\mathrm{grad}}$, hence $\boldsymbol{G}p \in \mathcal{V}_{\mathrm{grad}}$; similarly $\boldsymbol{D} : \mathcal{V}_{\mathrm{grad}} \to \mathcal{V}_{\mathrm{div}}$. With $\phi \equiv p$, the discrete version of (2.24) can be written as $\langle \boldsymbol{G}p, \boldsymbol{u}\rangle_{\mathcal{V}_{\mathrm{grad}}} = -\langle p, \boldsymbol{D}\boldsymbol{u}\rangle_{\mathcal{V}_{\mathrm{div}}}$. In matrix-vector inner product terminology, the notation of this relation can be further simplified to $(\boldsymbol{G}p)^*\boldsymbol{u} = -p^*\boldsymbol{D}\boldsymbol{u}$.

where $\boldsymbol{\Omega}$ is a diagonal matrix representing the sizes of the control volumes, $\boldsymbol{C}(\boldsymbol{u})$ is built from the convective flux contributions, and $\boldsymbol{V}$ contains the viscous, diffusive fluxes.

Similar to the discussion in Section 1.5, under the functional analytic constraints mentioned above, the evolution of the discrete energy $\boldsymbol{u}^*\boldsymbol{\Omega}\boldsymbol{u}$ is governed by[4]

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}(\boldsymbol{u}^*\boldsymbol{\Omega}\boldsymbol{u}) &= -\left(\boldsymbol{C}\boldsymbol{u}+\boldsymbol{V}\boldsymbol{u}+\boldsymbol{G}p\right)^*\boldsymbol{u}-\boldsymbol{u}^*\left(\boldsymbol{C}\boldsymbol{u}+\boldsymbol{V}\boldsymbol{u}+\boldsymbol{G}p\right)\\ &= -\boldsymbol{u}^*(\boldsymbol{C}+\boldsymbol{C}^*)\,\boldsymbol{u}-p^*\boldsymbol{G}^*\boldsymbol{u}-(\boldsymbol{G}^*\boldsymbol{u})^*p-\boldsymbol{u}^*(\boldsymbol{V}+\boldsymbol{V}^*)\,\boldsymbol{u}\\ &= p^*\boldsymbol{D}\boldsymbol{u}+(\boldsymbol{D}\boldsymbol{u})^*p-\boldsymbol{u}^*(\boldsymbol{V}+\boldsymbol{V}^*)\,\boldsymbol{u}\\ &= -\boldsymbol{u}^*(\boldsymbol{V}+\boldsymbol{V}^*)\,\boldsymbol{u}\le 0. \end{aligned} \tag{2.26}$$

In the derivation we have used that the convective part of the coefficient matrix is skew symmetric, and – equally important – that the discrete gradient is the adjoint of the discrete divergence (apart from a minus sign), i.e. $\boldsymbol{G}^*=-\boldsymbol{D}$, combined with $\boldsymbol{D}\boldsymbol{u}=0$. It follows that, for non-zero $\boldsymbol{u}$, the right-hand side of (2.26) is zero if and only if $\boldsymbol{V}+\boldsymbol{V}^*=\boldsymbol{0}$, i.e. the (discrete) energy is conserved if and only if the diffusion is turned off. With diffusion (i.e. for $\boldsymbol{V}+\boldsymbol{V}^*\neq\boldsymbol{0}$) the right-hand side is negative for all $\boldsymbol{u}\neq\boldsymbol{0}$. As a consequence, the energy of the discrete system (2.25) decreases in time when the symmetry-preserving discretization (2.23) is used: the semi-discrete system is stable, and a solution of (2.25) can be obtained on any grid (see e.g. Verstappen and Veldman 2003). Hence, there is no need to add artificial (numerical) damping in order to stabilize the spatial discretization. It also simplifies the task to keep time-integration methods for the solution of (2.25) stable (they preferably should also be energy preserving).

The above shows that there are close connections between the discretization of the convective terms, the continuity equation and the pressure gradient. In fact, once the convective terms have been discretized the discretization of the continuity equation is fixed, since it should make the diagonal coefficient in (2.22) equal to zero. After that, the discrete gradient follows from the adjoint of the discrete divergence. Note that diffusion appears to live a life of its own. It is symmetric, so the div- and grad-operators in the diffusive term are again each others adjoint, but (as far as we know) they are not related to the div- and grad-operators in continuity and pressure pradient.

Apart from the above discrete energy-preserving property, the differences between the discretizations of (2.19) and (2.19′) are small, but in certain conditions they can have consequences. In particular this holds when the solution possesses large gradients; use the conservation form (2.19′) in those cases, and always discretize in a symmetry-preserving way! A further theoretical fundament of the differences is hardly possible because of the non-linearity. In their book, Richtmyer and Morton (1967) pay relatively much attention to the complications that can arise from non-linearity.

[4]Physically, this derivation holds when the velocity along the boundary vanishes (e.g. solid walls). Periodic boundary conditions are also allowed, as long as the boundary terms cancel; no nett excitation (i.e. energy supply) from outside is allowed.

2.4 The Poisson equation for the pressure

2.4.1 Boundary conditions

As already remarked, the pressure in (2.16) has to be determined such that (2.15) is satisfied. This can be realized by combining these two equations. We can write (2.16) as

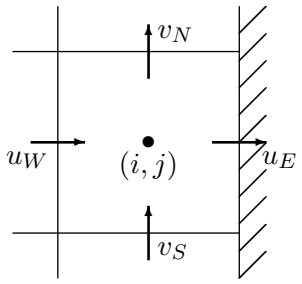
$$\boldsymbol{u}^{(n+1)} = \boldsymbol{u}^{(n)} + \delta t\, \boldsymbol{R}^{(n)} - \delta t \operatorname{grad} p^{(n+1)}, \tag{2.27}$$

and substitute this in (2.15), after which we obtain

$$\operatorname{div}\operatorname{grad} p^{(n+1)} = \operatorname{div}\,(\frac{\boldsymbol{u}^{(n)}}{\delta t} + \boldsymbol{R}^{(n)}). \tag{2.28}$$

This is the Poisson equation for the pressure. There are no boundary conditions directly available, since these only contain the velocity $\boldsymbol{u}$. This is no problem, however. We can first discretize (2.15) before we substitute the discrete version of (2.27)[5]. Note that after discretization of (2.15) the number of velocity unknowns equals the number of discrete momentum equations; and also the number of pressure unknowns is equal to the number of discrete continuity equations, since both are equal to the number of grid cells. Hence sufficient equations for all of the unknowns are available!

Thus the apparent lack of a boundary condition for the pressure is an artefact of the approach, which can easily be prevented. For a cell adjacent to a boundary of the computational domain this proceeds as follows.



Discrete conservation of mass (2.17′) gives

$$\frac{u_E^{(n+1)} - u_W^{(n+1)}}{h_x} + \frac{v_N^{(n+1)} - v_S^{(n+1)}}{h_y} = 0,$$

in which $u_E^{(n+1)} = 0$ according to the boundary condition (2.7). This is substituted first. Thereafter the other three velocities are transformed with the discretized (2.27).

In this way a discretization of (2.28) is created in which $p_{i,j}^{(n+1)}$, $p_{i,j+1}^{(n+1)}$, $p_{i,j-1}^{(n+1)}$ and $p_{i-1,j}^{(n+1)}$ appear, but not $p_{i+1,j}^{(n+1)}$ corresponding with a cell centre outside the computational domain. The result is a discretization of (2.28) in which no (visible) boundary conditions are required anymore. In fact they have been used the moment when $u_E = 0$ was substituted.

Let us introduce some notation to describe the above formally. In the discrete grid Ω two parts are distinguished: the grid points in the interior Ω^0 and those on the boundary Ω^Γ. The grid function for the velocity $\boldsymbol{u}$ is split similarly: $\boldsymbol{u}^0$ is defined in the grid points in Ω^0, while $\boldsymbol{u}^\Gamma$ is defined on the boundary Ω^Γ.

The discrete divergence operator as defined in (2.17′) is denoted by $\boldsymbol{D}$; the discrete gradient operator by $\boldsymbol{G}$. $\boldsymbol{D}$ acts on velocity components defined in grid points from $\Omega^0 \cup \Omega^\Gamma$, i.e. on

[5]This is a general strategy: discretize first, substitute the boundary conditions next, and only afterwards perform other operations on the equations (Veldman 1990). As soon as the discrete system of equations is OK, nothing 'dangerous' can happen anymore (what remains is 'just' a linear algebra problem).

$\boldsymbol{u}^0$ and $\boldsymbol{u}^\Gamma$ (since $\operatorname{div} \boldsymbol{u} = 0$ is only discretized in interior points). We split the divergence operator in two parts

$$\boldsymbol{D} = \boldsymbol{D}^0 + \boldsymbol{D}^\Gamma, \tag{2.29}$$

where $\boldsymbol{D}^0$ acts on values of $\boldsymbol{u}$ in the interior of Ω (i.e. on $\boldsymbol{u}^0$), and $\boldsymbol{D}^\Gamma$ corresponds with $\boldsymbol{u}^\Gamma$. The above method can now be written as follows.

First write the discretized version of (2.15) as

$$\boldsymbol{D}^0 \boldsymbol{u}^{(n+1)} = -\boldsymbol{D}^\Gamma \boldsymbol{u}^{(n+1)}, \tag{2.30}$$

in which the right-hand side can be determined from the boundary condition for $\boldsymbol{u}$. Discretize (2.27) in the interior points from Ω^0

$$\boldsymbol{u}^{(n+1)} = \boldsymbol{u}^{(n)} + \delta t\, \boldsymbol{R}^{(n)} - \delta t\, \boldsymbol{G}\, p^{(n+1)}, \tag{2.31}$$

where $\boldsymbol{R}^{(n)}$ is the discrete version of $\boldsymbol{R}^{(n)}$. Now substitute (2.31) in the left-hand side of (2.30) and note that only interior points occur for $\boldsymbol{u}$. The result can be written as

$$\boldsymbol{D}^0 \boldsymbol{G} p^{(n+1)} = \boldsymbol{D}^0 \left(\frac{\boldsymbol{u}^{(n)}}{\delta t} + \boldsymbol{R}^{(n)}\right) + \boldsymbol{D}^\Gamma \frac{\boldsymbol{u}^{(n+1)}}{\delta t}. \tag{2.32}$$

In the left-hand side we have one equation for each cell, corresponding with one unknown p per cell. There are no values of p belonging to cell centres outside Ω.

The operator in the left-hand side of (2.32) is singular, as a pressure $\tilde{p}$ that is constant satisfies $\boldsymbol{G}\tilde{p} = 0$, with which also $\boldsymbol{D}^0\boldsymbol{G}\tilde{p} = 0$. In turn $p =$ Constant is the only solution of $\boldsymbol{D}^0\boldsymbol{G}p = 0$. This is a consequence of the choice of the computational grid; in Section 2.2 we have discussed this extensively. The fact that the operator in the left-hand side of (2.32) is singular, does imply that there need not be a solution for each right-hand side. The latter has to satisfy a compatibility relation: it has to be perpendicular to the kernel of $(\boldsymbol{D}^0\boldsymbol{G})^*$. This leads to a condition for the boundary condition of $\boldsymbol{u}$ on Γ: the discrete nett mass flow through the boundary Γ has to be zero (i.e. we must have discrete conservation of mass over the whole domain Ω). This restriction can usually be satisfied in a discrete sense, since it is satisfied in the continuous case. This compatibility relation corresponds with the relation that has to be satisfied by the boundary condition in a Neumann problem for the Poisson equation.

Remark In the literature, the combination $\boldsymbol{u}^{(n)} + \delta t\, \boldsymbol{R}^{(n)}$ in (2.27) is often referred to as an auxiliary velocity. Some people even try to derive boundary conditions for it, as these seem to be necessary in the right-hand side of (2.28). The correction with $\delta t \operatorname{grad} p^{(n+1)}$ is called the projection onto the space of divergence-free functions, whence the approach is often called a projection method. This is a nice geometric interpretation, but there is no necessity to use such fancy words. It is just a straightforward (linear algebra) combination of the original discrete equations (2.15) and (2.16).

Equivalent alternative

In the literature the problem of the missing boundary condition for p is often solved in a different way: sometimes this is equivalent to the above approach, sometimes it is not. An

equivalent way is to consider the x-component of (2.16) on the right-hand boundary. After substitution of the boundary condition for $\boldsymbol{u}$ we have

$$\frac{\partial}{\partial n}p^{(n+1)} = \boldsymbol{n}\cdot[\boldsymbol{R}^{(n)} - \frac{\boldsymbol{u}^{(n+1)}-\boldsymbol{u}^{(n)}}{\delta t}], \tag{2.33}$$

in which the normal on the wall is indicated with $\boldsymbol{n}$. This equation can be discretized as in (2.31)

$$\boldsymbol{n}\cdot\boldsymbol{G}\,p^{(n+1)} = \boldsymbol{n}\cdot[\boldsymbol{R}^{(n)} - \frac{\boldsymbol{u}^{(n+1)}-\boldsymbol{u}^{(n)}}{\delta t}]. \tag{2.33$'$}$$

The discretization of $\boldsymbol{R}^{(n)}$ poses problems, because more information from outside the computational domain is required (the 'normal'-velocity one mesh outside the wall). The Poisson equation that is solved is

$$\boldsymbol{D}\boldsymbol{G}p^{(n+1)} = \boldsymbol{D}(\frac{\boldsymbol{u}^{(n)}}{\delta t} + \boldsymbol{R}^{(n)}). \tag{2.34}$$

Here $\boldsymbol{R}^{(n)}$ features as well. We will show that when the $\boldsymbol{R}^{(n)}$ in (2.33′) and (2.34) are treated in the same way, this does not have consequences for the solution of $p^{(n+1)}$ in the interior of Ω. It does for values of $p^{(n+1)}$ outside Ω, therefore one has to be careful when the pressure p on the boundary of Ω is obtained through interpolation. Consider both sides of (2.34)

$$\boldsymbol{D}\boldsymbol{G}p^{(n+1)} \;=\; \boldsymbol{D}^0\boldsymbol{G}p^{(n+1)} + \boldsymbol{D}^\Gamma\boldsymbol{G}p^{(n+1)} \;=\; \boldsymbol{D}^0\boldsymbol{G}p^{(n+1)} + \boldsymbol{D}^\Gamma[\boldsymbol{R}^{(n)} - \frac{\boldsymbol{u}^{(n+1)}-\boldsymbol{u}^{(n)}}{\delta t}],$$

and

$$\boldsymbol{D}(\frac{\boldsymbol{u}^{(n)}}{\delta t} + \boldsymbol{R}^{(n)}) \;=\; \boldsymbol{D}^0(\frac{\boldsymbol{u}^{(n)}}{\delta t} + \boldsymbol{R}^{(n)}) + \boldsymbol{D}^\Gamma\frac{\boldsymbol{u}^{(n)}}{\delta t} + \boldsymbol{D}^\Gamma\boldsymbol{R}^{(n)},$$

respectively. Finally, observe that substitution of these relations in (2.34) leads to (2.32).

Despite the correct approach is known for more than forty years, there has been a lot of unclarity in the literature; see e.g. Roache (1988). For a comprehensive discussion of this subject we refer to Gresho and Sani (1987).

2.4.2 Treatment of div $u^{(n)}$

When the Poisson equation (2.28) is solved exactly then $\operatorname{div}\boldsymbol{u}^{(n+1)} = 0$. We could substitute this in the right-hand side of (2.28) on the next time level (i.e. set $\operatorname{div}\boldsymbol{u}^{(n)} = 0$). However, in reality (2.28) is not solved exactly (at most machine accuracy can be reached; we would prefer to stop an iterative method earlier), hence an error is made. When $\operatorname{div}\boldsymbol{u}^{(n)} = 0$ is substituted the danger exists that these errors accumulate. Error accumulation can be prevented by leaving the term $\operatorname{div}\boldsymbol{u}^{(n)}$ in the equation. The functioning of this correcting term can be seen as follows (Hirt and Harlow 1967):

Start taking the divergence of (2.16)

$$\frac{1}{\delta t}(\operatorname{div}\boldsymbol{u}^{(n+1)} - \operatorname{div}\boldsymbol{u}^{(n)}) + \operatorname{div}\operatorname{grad}p^{(n+1)} = \operatorname{div}\boldsymbol{R}^{(n)}. \tag{2.35}$$

If we were to solve (2.28) with $\operatorname{div}\boldsymbol{u}^{(n)} = 0$ substituted, i.e.

$$\operatorname{div}\operatorname{grad}p^{(n+1)} = \operatorname{div}\boldsymbol{R}^{(n)}, \tag{2.36}$$

then after substitution in (2.35) this leads to

$$\frac{1}{\delta t}(\operatorname{div} \boldsymbol{u}^{(n+1)} - \operatorname{div} \boldsymbol{u}^{(n)}) = 0. \tag{2.37}$$

This is a discrete version of $\frac{\partial}{\partial t}\operatorname{div} \boldsymbol{u} = 0$, which analytically is equivalent is with $\operatorname{div} \boldsymbol{u} = 0$ (at $t = 0$ we start with $\operatorname{div} \boldsymbol{u} = 0$). However, numerical error accumulation is possible, because an error made in computing $\operatorname{div} \boldsymbol{u}^{(n)}$ is fully visible in $\operatorname{div} \boldsymbol{u}^{(n+1)}$, through (2.37). Such errors are made when (2.36) is not solved exactly, but with an error $\varepsilon^{(n+1)}$:

$$\operatorname{div} \operatorname{grad} p^{(n+1)} = \operatorname{div} \boldsymbol{R}^{(n)} + \varepsilon^{(n+1)}. \tag{2.38}$$

Substitution in (2.35) yields

$$\operatorname{div} \boldsymbol{u}^{(n+1)} = \operatorname{div} \boldsymbol{u}^{(n)} - \delta t\, \varepsilon^{(n+1)}.$$

When, for example, systematically the same error ϵ is made in solving (2.36), then at a fixed moment in time $t = n\,\delta t$ we have

$$\operatorname{div} \boldsymbol{u}^{(n)} = \operatorname{div} \boldsymbol{u}^{(0)} - n\,\delta t\,\varepsilon = \operatorname{div} \boldsymbol{u}^{(0)} - t\,\varepsilon.$$

Letting $\delta t \to 0$ gives no improvement. A fortiori, when we let $t \to \infty$, e.g. when we are interested in the steady limit, then even $\operatorname{div} \boldsymbol{u}^{(n)} \to \infty$. Moreover, when the initial velocity field is not divergence free, this is not corrected!

If, on the other hand, (2.28) is solved instead of (2.36), then from (2.35) it follows that $\operatorname{div} \boldsymbol{u}^{(n+1)} = 0$ (the discrete version of $\operatorname{div} \boldsymbol{u} = 0$). A numerical error like in (2.38) now leads to

$$\operatorname{div} \boldsymbol{u}^{(n+1)} = -\delta t\, \varepsilon^{(n+1)},$$

such that no accumulating effect of these errors is present. Moreover, $\delta t \to 0$ leads to $\operatorname{div} \boldsymbol{u}^{(n)} \to 0$.

The above reasoning[6] is also valid for the spatially discretized version (2.30)-(2.32). When the approach without error accumulation is used, less stringent requirements have to be imposed on solving the Poisson equation. Its solution process thus becomes cheaper.

2.4.3 Pressure iteration

Above we have seen how we can solve the problem of the missing boundary condition for p in a safe manner. The formulas become somewhat more complicated in their presentation, which will be neutralized from now on by restricting ourselves to the homogeneous condition $\boldsymbol{u} = 0$ on Γ. The equations to be solved become in this case

$$\boldsymbol{D}^0\, \boldsymbol{u}^{(n+1)} = 0, \tag{2.39}$$

$$\boldsymbol{u}^{(n+1)} + \delta t\, \boldsymbol{G} p^{(n+1)} = \boldsymbol{u}^{(n)} + \delta t\, \boldsymbol{R}^{(n)}. \tag{2.40}$$

[6]This approach also fits in the strategy from the footnote on page 60: once (2.15)+(2.16) have been discretized, don't try to be smart but stick to linear algebra.

Instead of this system we can also solve (2.40) combined with the Poisson equation

$$\boldsymbol{D}^0\boldsymbol{G}p^{(n+1)} = \boldsymbol{D}^0(\frac{\boldsymbol{u}^{(n)}}{\delta t} + \boldsymbol{R}^{(n)}). \tag{2.41}$$

The Poisson equation can be solved in many ways, for example with a direct method. In recent years many 'Fast Poisson Solvers' have been developed. One has to be aware that the equation is singular - the pressure level is not fixed. Also, often iterative techniques are being used. Because we will encounter these again, in disguise, in the next sections, we will spend some attention to them here.

We start with the Jacobi method with relaxation (JOR; see Appendix A.2.1) for solving (2.41). The operator symbols are to be interpreted as matrices now; let Λ be the diagonal of $\boldsymbol{D}^0\boldsymbol{G}$. Following (A.18c), the JOR method for solving (2.41) becomes

$$(p^{(n+1)})^{(k+1)} = (p^{(n+1)})^{(k)} - \omega\Lambda^{-1}\left\{\boldsymbol{D}^0\boldsymbol{G}(p^{(n+1)})^{(k)} - \boldsymbol{D}^0(\frac{\boldsymbol{u}^{(n)}}{\delta t} + \boldsymbol{R}^{(n)})\right\}, \tag{2.42}$$

where ω is the relaxation factor and k the iteration index.

An at first sight different technique results when (2.39) and (2.40) are being used. Suppose an estimate $(p^{(n+1)})^{(k)}$ for the pressure is available, then (2.40) yields an estimate for the velocity

$$(\boldsymbol{u}^{(n+1)})^{(k+1)} = (\boldsymbol{u}^{(n)} + \delta t\, \boldsymbol{R}^{(n)}) - \delta t\, \boldsymbol{G}(p^{(n+1)})^{(k)}. \tag{2.43}$$

In general this will not satisfy (2.39) and therefore a new pressure estimate is determined through

$$(p^{(n+1)})^{(k+1)} = (p^{(n+1)})^{(k)} + \beta\, \boldsymbol{D}^0(\boldsymbol{u}^{(n+1)})^{(k+1)}, \tag{2.44}$$

where β is a relaxation parameter. This iteration process does not contain a Poisson equation, but ... combine (2.43) with (2.44). Then we have

$$(p^{(n+1)})^{(k+1)} = (p^{(n+1)})^{(k)} - \beta\, \delta t\, \boldsymbol{D}^0\boldsymbol{G}(p^{(n+1)})^{(k)} + \beta\boldsymbol{D}^0(\boldsymbol{u}^{(n)} + \delta t\, \boldsymbol{R}^{(n)}). \tag{2.45}$$

And this is the same as (2.42), when the relaxation parameters ω and β are being chosen such that

$$\omega = \lambda_i\, \beta\, \delta t,$$

where λ_i is the diagonal element of the Poisson matrix in the i^{th} grid point. Therefore both methods discussed are in fact identical when grid-point dependent values for ω and β are allowed. This is a non-traditional idea, but nevertheless can work extremely well (Botta and Veldman 1982).

2.5 The steady Navier–Stokes equations

2.5.1 Discrete formulation

In the preceding pages an explicit method has been presented to solve the unsteady Navier–Stokes equations (2.12)+(2.13). This method can also be used to determine the steady solution. But, of course, there are other options. In this section we will discuss two methods to

solve

$$\begin{aligned} \operatorname{div} \boldsymbol{u} &= 0, && (2.46)\\ \operatorname{grad} p &= \boldsymbol{R}(\boldsymbol{u}), && (2.47) \end{aligned}$$

where $\boldsymbol{R}(\boldsymbol{u})$ is an abbreviation, given in (2.14), for the convective and diffusive terms.

After discretization and linearization of (2.46) a system of equations is created with the following matrix-vector structure

$$\begin{pmatrix} -\boldsymbol{M} & \boldsymbol{G} \\ \boldsymbol{D}^0 & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{u} \\ p \end{pmatrix} = \begin{pmatrix} (\boldsymbol{R}-\boldsymbol{M})(\boldsymbol{u}) \\ 0 \end{pmatrix}, \tag{2.48}$$

where $\boldsymbol{M}$ is a linearized version of $\boldsymbol{R}$ (e.g. obtained via Newton linearization). The matrix in the left-hand side is singular (because the pressure is determined up to a constant). More problematic is the zero submatrix in the right-bottom corner which is present because the pressure does not appear in the continuity equation. Such a matrix is called a *saddle-point* matrix.

Direct inversion of the left-hand-side matrix in (2.48) is not always possible, as the system is very large and memory requirements can become prohibitive. Yet, for smaller-sized problems direct methods provide the fastest way of solving the system. Further, standard iterative methods are not applicable because of the zero submatrix. Thus there is a need for more efficient solvers, often a blend between direct and iterative methods. For instance, at RUG research is going on to solve the full system (2.48) with preconditioned conjugate gradient methods (Botta and Wubs 1999; Wubs and Thies 2011).

2.5.2 Artificial compressibility

Because solving the compressible equations does have its advantages, it may make sense to approximate the incompressible equations with the compressible ones. Inspired by (2.9), Chorin (1967) introduced an artificial gas model $p = c^2\,\rho$, and substituted it in the continuity equation (after normalizing ρ at 1). In this way one obtains

$$\frac{\partial p}{\partial t} + c^2 \operatorname{div} \boldsymbol{u} = 0. \tag{2.49}$$

This equation does not have a physical meaning as far as it concerns the time-dependent behaviour. But when a limit-solution exists for $t \to \infty$, then $\partial p/\partial t = 0$ after which (2.49) implies the first equation in (2.46). To the second equation in (2.46) we add the 'normal' time derivative (other choices are possible). After discretization in space and time (explicit) the following algorithm is obtained

$$\begin{aligned} p^{(k+1)} &= p^{(k)} - c^2\,\delta t\,\boldsymbol{D}^0\boldsymbol{u}^{(k)}, && (2.50)\\ \boldsymbol{u}^{(k+1)} &= \boldsymbol{u}^{(k)} + \delta t\,\boldsymbol{R}^{(k)} - \delta t\,\boldsymbol{G}p^{(k)}. && (2.51) \end{aligned}$$

By comparing with (2.43)+(2.44), it is observed that the above method is just an unsteady version hereof in which $\beta = -c^2\delta t$.

2.5.3 SIMPLE

Spalding & Patankar (see e.g. Patankar 1980) have developed software packages for flow simulation (TEACH, CHAMPION, PHOENICS) in which use is made of the SIMPLE algorithm (or variants hereof) to solve the incompressible equations of motion. SIMPLE stands for Semi-Implicit Method for Pressure Linked Equations. It is one of the most-used algorithms to solve the steady Navier–Stokes equations, therefore we describe it below.

Start with the discrete version of (2.46) and (2.47) and assume that an estimate for the pressure $p^{(k)}$ is available. Then solve

$$\boldsymbol{R}(\boldsymbol{u}^{(k)}) = \boldsymbol{G}p^{(k)}. \tag{2.52}$$

In general $\boldsymbol{u}^{(k)}$ will not be divergence free. Therefore introduce a pressure correction δp, with a corresponding velocity correction $\delta\boldsymbol{u}$, in other words

$$\boldsymbol{R}\,(\boldsymbol{u}^{(k)} + \delta\boldsymbol{u}) = \boldsymbol{G}\,(p^{(k)} + \delta p).$$

From this relation $\delta\boldsymbol{u}$ is solved *approximately*: $\delta\boldsymbol{u} = \boldsymbol{G}\delta p$, and require next that $\boldsymbol{D}^0(\boldsymbol{u}^{(k)} + \delta\boldsymbol{u}) = 0$, hence

$$\boldsymbol{D}^0\boldsymbol{G}\,\delta p = -\boldsymbol{D}^0\boldsymbol{u}^{(k)}. \tag{2.53}$$

Solving this equation provides a new estimate for the pressure

$$p^{(k+1)} = p^{(k)} + \delta p; \tag{2.54}$$

often here relaxation is applied. Finally we can return to (2.52).

In (2.52) as well as (2.53) a system of equations has to be solved. This can be done with a direct method, but also iteratively. This iteration process can then be combined with the iteration process in (2.52)–(2.54). As an example (2.52) and (2.53) are solved with JOR, and apply one JOR iteration per k-iteration. Instead of (2.52) we obtain

$$\boldsymbol{u}^{(k+1)} = \boldsymbol{u}^{(k)} + \omega_u\,\{\boldsymbol{R}(\boldsymbol{u}^{(k)}) - \boldsymbol{G}p^{(k)}\}. \tag{2.55}$$

Next, (2.53) is solved approximately with one JOR iteration, while the initial guess $\delta p = 0$ is chosen

$$\delta p = -\omega_p\,\boldsymbol{D}^0\boldsymbol{u}^{(k)}. \tag{2.56}$$

The iterative process (2.55)+(2.56) is related to (2.50)+(2.51). The choice $\omega_u = \delta t$ and $\omega_p = c^2\delta t$ makes both processes identical. From the optimal δt following from a stability study of (2.50)+(2.51) a good choice for both ω's can be deduced.

Simply spoken one can say that in SIMPLE the time integration of the unsteady approach (e.g. MAC) and the iterations of the Poisson equation have been merged together. Therefore it is somewhat peculiar to apply a SIMPLE-like approach per time step in unsteady problems, as is often done.

In the preceding pages some frequently used iterative methods have been discussed. Closer inspection has revealed that they are intimately related.

2.6 Discretization - implicit

When fully implicit (backward Euler) discretized in time, the incompressible Navier–Stokes equations read

$$\operatorname{div} \boldsymbol{u}^{(n+1)} = 0, \tag{2.57}$$

$$\frac{1}{\delta t}(\boldsymbol{u}^{(n+1)} - \boldsymbol{u}^{(n)}) + \operatorname{grad} p^{(n+1)} - \boldsymbol{R}^{(n+1)} = 0. \tag{2.58}$$

Combination of these two equations gives the Poisson equation

$$\operatorname{div}\operatorname{grad} p^{(n+1)} - \operatorname{div} \boldsymbol{R}^{(n+1)} = \operatorname{div} \frac{\boldsymbol{u}^{(n)}}{\delta t}. \tag{2.59}$$

After discretization in spatial direction, where the boundary condition $\boldsymbol{u} = \boldsymbol{u}^\Gamma$ on Γ has already been substituted as in Section 2.4.1, we obtain

$$\boldsymbol{D}^0\, \boldsymbol{u}^{(n+1)} = -\boldsymbol{D}^\Gamma\, \boldsymbol{u}^\Gamma, \tag{2.60}$$

$$\boldsymbol{u}^{(n+1)} - \delta t\, \boldsymbol{R}(\boldsymbol{u}^{(n+1)}) + \delta t\, \boldsymbol{G} p^{(n+1)} = \boldsymbol{u}^{(n)}. \tag{2.61}$$

The discrete Poisson equation reads

$$\boldsymbol{D}^0 \boldsymbol{G} p^{(n+1)} - \boldsymbol{D}^0 \boldsymbol{R}(\boldsymbol{u}^{(n+1)}) = \boldsymbol{D}^0 \frac{\boldsymbol{u}^{(n)}}{\delta t} + \boldsymbol{D}^\Gamma \frac{\boldsymbol{u}^\Gamma}{\delta t} \tag{2.62}$$

Remark 1 For the following presentation it is not relevant which time-integration method for $\boldsymbol{R}$ is applied. Despite the time index has been written as $n+1$, $\boldsymbol{R}^{(n+1)}$ may also contain contributions from earlier time levels, such that e.g. the method of Crank-Nicolson is also covered.

Remark 2 In (2.33) and further an alternative way to treat the boundary condition for the Poisson equation has been described. For explicit time-integration it could be proved that this is equivalent; for implicit time-integration this can no longer be proved, as the equations at the new time level are not solved exactly.

Remark 3 Due to the time derivative, solving the momentum equation for the velocity is usually cheap compared to solving the Poisson equation for the pressure. Therefore, the price for calculating an implicit time step is not much higher than that for an explicit time step, where the pressure is calculated implicitly anyway. Hence, at almost the same cost larger time steps can be taken (but watch out for the accuracy). In algorithms for computing compressible flow (not discussed in these lecture notes) this balance is different.

2.6.1 Linearization

The term $\boldsymbol{R}^{(n+1)}$ contains the discrete convective and diffusive terms; the former are non-linear. We will next present some methods for treating this non-linearity. As an example we will treat a generic term $u\phi_x$, which we write after discretization as

$$u^{(n+1)} \phi_x^{(n+1)}. \tag{2.63}$$

We leave open whether a central or an upwind discretization has been used. This term will be linearized: either with respect to the old time level n, or with respect to the new level $n+1$.

a) *Linearization around the old time level:*
(2.63) is approximated by

$$u^{(n)}\phi_x^{(n+1)} \tag{2.64}$$

where we make an error $O(\delta t)$. When the remainder of the method possesses a similar discretization error everything nicely fits together.

b) *Newton-linearization around the old time level:*
Write

$$u^{(n+1)} = u^{(n)} + \delta u \quad \text{and} \quad \phi_x^{(n+1)} = \phi_x^{(n)} + \delta\phi_x,$$

then δu and $\delta\phi_x$ are of the order $O(\delta t)$. Substitution in (2.63) yields

$$\begin{aligned} u^{(n+1)}\phi_x^{(n+1)} &= u^{(n)}\phi_x^{(n)} + u^{(n)}\,\delta\phi_x + \delta u\,\phi_x^{(n)} + O(\delta t^2) \\ &= u^{(n)}\phi_x^{(n+1)} + u^{(n+1)}\phi_x^{(n)} - u^{(n)}\phi_x^{(n)} + O(\delta t^2). \end{aligned} \tag{2.65}$$

The expression in the right-hand side of (2.65) is linear in the variables at the new time level and possesses a discretization error $O(\delta t^2)$. Thus second-order time accuracy can be achieved at the cost of only one linear solve of the system (2.57)+(2.58).

c) *Linearization around the new time level:*
At the cost of introducing an additional iteration process the linearization around the new time level can be performed. This proceeds similar to a) or b), but now no additional discretization error is being made. E.g., the version analogous to a) reads

$$(u^{(n+1)})^{(k)}(\phi_x^{(n+1)})^{(k+1)}, \tag{2.66}$$

where k is the counter of the additional iteration process. Often such an additional process has been introduced already because of other reasons, and the iteration (2.66) can be included 'on the fly'.

2.6.2 Pressure correction

An orderly way to solve the implicit system is the *pressure-correction* method. A good explanation of this method, including reflections on accuracy and stability, has been given by Van Kan (1986). In this method first a 'predictor' velocity field $\boldsymbol{u}^*$ is determined by solving the following implicit equation

$$\frac{\boldsymbol{u}^* - \boldsymbol{u}^{(n)}}{\delta t} + \operatorname{grad} p^{(n)} - \boldsymbol{R}^* = 0. \tag{2.67}$$

Next the pressure-correction is determined from the Poisson equation

$$\operatorname{div}\operatorname{grad}(p^{(n+1)} - p^{(n)}) = \operatorname{div}\left(\frac{\boldsymbol{u}^*}{\delta t}\right), \tag{2.68}$$

which is equivalent with (compare (2.28) and (2.59))

$$\operatorname{div}\operatorname{grad} p^{(n+1)} = \operatorname{div}\left(\frac{\boldsymbol{u}^{(n)}}{\delta t} + \boldsymbol{R}^*\right).$$

Finally, with this newly calculated pressure field, the new velocity field is computed from a 'simple' substitution

$$\boldsymbol{u}^{(n+1)} = \boldsymbol{u}^* - \delta t \,\mathrm{grad}\,(p^{(n+1)} - p^{(n)}), \tag{2.69}$$

where (2.68) guarantees that $\boldsymbol{u}^{(n+1)}$ is divergence-free.

By combining (2.67) with (2.69) it follows that effectively

$$\frac{\boldsymbol{u}^{(n+1)} - \boldsymbol{u}^{(n)}}{\delta t} + \mathrm{grad}\, p^{(n+1)} = \boldsymbol{R}^*$$

has been solved. This is not quite the desired equation (2.58), but the difference between $\boldsymbol{R}^*$ and $\boldsymbol{R}^{(n+1)}$ is small. When e.g. the Crank-Nicolson method is used (instead of backward Euler), the final $\boldsymbol{u}^{(n+1)}$ is $O(\delta t^2)$ accurate, as shown by Van Kan (1986). Of course, if this difference is deemed too large, a second corrector step can be made as is done e.g. in the closely-related PISO method proposed by Issa (1986) (apart from a slightly different treatment of the diagonal of $\boldsymbol{R}$).

Alternatives?

But why do it neatly when you can also make a mess of it? The equations (2.60)–(2.62) resemble (2.39)–(2.41) where an explicit time integration has been applied. The difference lies in the treatment of $\boldsymbol{R}$, which now has to be evaluated at the new time level. This can be done, for example, by letting $\boldsymbol{R}$ evolve with the k-iteration in the iterative methods from Section 2.4.3. In the formulas (2.42), (2.43) and (2.45) $\boldsymbol{R}^{(n)}$ is replaced by $\boldsymbol{R}(\boldsymbol{u}^{(n+1)})^{(k)}$. As an example (2.43)+(2.45) becomes

$$(\boldsymbol{u}^{(n+1)})^{(k+1)} = \boldsymbol{u}^{(n)} + \delta t\,[\boldsymbol{R}(\boldsymbol{u}^{(n+1)})^{(k)} - \boldsymbol{G}(p^{(n+1)})^{(k)}], \tag{2.70}$$

$$(p^{(n+1)})^{(k+1)} = (p^{(n+1)})^{(k)} - \beta\,\delta t\,\boldsymbol{D}^0\boldsymbol{G}(p^{(n+1)})^{(k)} + \beta\,\boldsymbol{D}^0[\boldsymbol{u}^{(n)} + \delta t\,\boldsymbol{R}(\boldsymbol{u}^{(n+1)})^{(k+1)}]. \tag{2.71}$$

The treatment in (2.70) is related to the predictor-corrector method. The factor δt outside the square brackets allows that 2 to 3 iterations suffice (each iteration a factor $O(\delta t)$ is gained).

When we want to keep the Poisson equation visible, we can e.g. combine (2.70) with

$$\boldsymbol{D}^0\boldsymbol{G}(p^{(n+1)})^{(k+1)} = \boldsymbol{D}^0\boldsymbol{R}(\boldsymbol{u}^{(n+1)})^{(k+1)} + \boldsymbol{D}^0\frac{\boldsymbol{u}^{(n)}}{\delta t}.$$

It is tempting to solve this equation iteratively. In this way a three-fold nested iterative process is created. Those who know whether the whole will converge may speak up ...

When the $\boldsymbol{R}$-part in (2.70) is moved to the left-hand side and only treat the $\boldsymbol{G}$-part in a predictor-corrector way, then we obtain an unsteady variation of (2.52):

$$(\boldsymbol{u}^{(n+1)})^{(k+1)} - \delta t\,\boldsymbol{R}(\boldsymbol{u}^{(n+1)})^{(k+1)} = \boldsymbol{u}^{(n)} - \delta t\,\boldsymbol{G}(p^{(n+1)})^{(k)}.$$

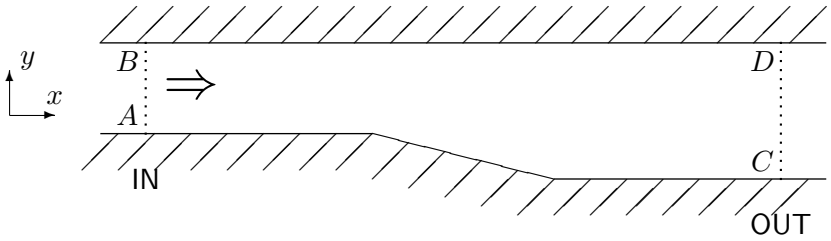
It is also possible to place an alternating part of $\boldsymbol{R}$ in the left-hand side, as in the alternating direction implicit (ADI) methods. In short, with some phantasy many methods can be invented to determine the solution at the new time level. An advice: keep the method simple and analyzable. This also reduces the chance on programming errors, and increases the

readability and maintainability of the programme. Remind that a more complicated method (and computer code) might solve the system of equations faster, but this activity is only a limited portion of the complete simulation (including pre- and postprocessing), such that a gain in CPU time for the solver does not have much impact on the time required for the whole problem.

2.7 In- and outflow conditions

The choice of boundary conditions at in- and outflow openings is an art in itself. Recently two minisymposia have been held that were devoted exclusively to this problem. A report of the findings has been given by Sani and Gresho (1994), and is summarized by them as: "... perhaps nowhere else do theory and practice seem to clash so much."

The treatment of in- and outflow conditions will be illustrated by a two-dimensional channel flow as shown in the figure.



Inflow conditions

The inflow boundary usually does not lead to too many problems. Often both velocity components, u and v, are prescribed. Thus, as for a solid wall, no boundary condition for the pressure is required. It is preferable to let the values of u and v be consistent with the boundary conditions next to the inlet. E.g. in the shown configuration one has $u = v = 0$ in the points A and B. Further, the continuity equation fixes $\partial v/\partial y$ because

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0.$$

At the wall in the points A and B the no-slip condition $u = 0$ induces $\partial u/\partial x = 0$, hence in these points $\partial v/\partial y = 0$. When the inflow profile is inconsistent, the discrete solution near A and B becomes irregular (but usually the 'damage' is not that big and only local).

An alternative is to prescribe a pressure difference across the channel, in combination with e.g. $v = 0$ and $\partial u/\partial x = 0$ at inflow and outflow (but see the discussion below).

Outflow conditions

The outflow boundary is much more problematic. A central role is being played by the normal stress and the shear stress along the outflow boundary. As a reminder (see Section 2.1), for incompressible flow the normal stress is given by

$$\boldsymbol{\sigma_n} \equiv \boldsymbol{\sigma} \cdot \boldsymbol{n} = -p + 2\mu \frac{\partial u_n}{\partial n},$$

and the tangential stress by

$$\boldsymbol{\sigma_t} \equiv \boldsymbol{\sigma} \cdot \boldsymbol{t} = \mu \, (\frac{\partial u_n}{\partial t} + \frac{\partial u_t}{\partial n}).$$

Here $u_n = \boldsymbol{u} \cdot \boldsymbol{n}$ is the velocity in the direction of the normal and $u_t = \boldsymbol{u} \cdot \boldsymbol{t}$ the velocity in tangential direction.

When at an outflow boundary the normal velocity u_n is prescribed then a boundary layer may be created (or even worse: large wiggles), as we have seen in the discussion of the convection-diffusion equation. It is therefore better to replace this Dirichlet condition by a homogeneous Neumann condition: $\partial u_n/\partial n = 0$. Physically this means that there is no viscous normal stress contribution. The continuity equation now yields $\partial u_t/\partial t = 0$, hence $u_t = 0$, with which two boundary conditions are available. Sani and Gresho (1994) comment this method as: "$\partial u_n/\partial n = 0$ is illegal ('hopelessly ill-posed') because of insufficient information; the system is underdetermined, resulting (in general) in an infinite number of solutions. In addition, in 2D it is overly restrictive on u_t, causing $u_t = 0$ regardless of the actual boundary condition applied to u_t." When additionally a condition on the pressure is prescribed, e.g. $p = 0$, then this is "illegal (ill-posed) because of too much information; the system is overdetermined and no solution exists in general." However, in practice the latter choice is not that bad!

Another way is to set $\partial^2 u_n/\partial n^2 = 0$ at the outflow boundary, and to discretize $\partial u_n/\partial n$ in an upwind way (central discretization will yield the same result, though). In this case no values of u_n outside the domain are required. A consistent condition for the tangential velocity is $\partial u_t/\partial n = 0$. Again a pressure condition 'should' be added.

A mathematically better way is to prescribe the normal stress $\boldsymbol{\sigma_n} = 0$. Sani and Gresho write: "[This condition] is well-posed but may not always be useful." This boundary condition is a natural condition and is often being used in finite-element formulations. In the finite-element world it is logical to prescribe additionally the tangential stress $\boldsymbol{\sigma_t} = 0$.

Sani and Gresho (1994) have studied much more candidate outflow conditions. There is no clear winner. A safe strategy is to set the outflow position as downstream as possible: "Nothing interesting should be happening at such a boundary; otherwise the boundary is in the wrong place."

2.8 References

A. Arakawa (1966) Computational design for long-term numerical integration of the equations of fluid motion: two-dimensional incompressible flow. *J. Comput. Phys.* 1, 119–143.

E.F.F. Botta and A.E.P. Veldman (1982) On local relaxation methods and their application to convection-diffusion equations. *J. Comput. Phys.* 48, 127–149.

E.F.F. Botta and F.W. Wubs (1999) Matrix Renumbering ILU: an effective algebraic multilevel ILU preconditioner for sparse matrices. *SIAM J. Matrix Anal. Appl.* 20, 1007–1026.

A.J. Chorin (1967) A numerical method for solving incompressible viscous flow problems. *J. Comput. Phys.* 2, 12–26.

P.M. Gresho and R.L. Sani (1987) On pressure boundary conditions for the incompressible Navier–Stokes equations. *Int. J. Num. Meth. Fluids* 7, 1111–1145.

F.H. Harlow and J.E. Welsh (1965) Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *Phys. Fluids* 8, 2182–2189.

C.W. Hirt and F.H. Harlow (1967) A general corrective procedure for the numerical solution of initial-value problems. *J. Comput. Phys.* 2, 114–119.

R.I. Issa (1986) Solution of the implicitly discretised fluid flow equations by operator-splitting. *J. Comput. Phys.* 62, 40–65.

J. van Kan (1986) A second-order accurate pressure-correction scheme for viscous incompressible flow. *SIAM J. Sci. Stat. Comput.* 7, 870–891.

S.V. Patankar (1980) *Numerical Heat Transfer and Fluid Flow.* Hemisphere–McGraw Hill.

C.M. Rhie and W.L. Chow (1983) Numerical study of the turbulent flow past an airfoil with trailing edge separation. *AIAA J.* 21, 1525–1532.

R.D. Richtmyer and K.W. Morton (1967) *Difference Methods for Initial Value Problems.* John Wiley.

P. Roache (1988) A comment on the paper "Finite difference methods for the Stokes and Navier–Stokes equations" by J.C. Strikwerda. *Int. J. Num. Meth. Fluids* 8, 1459–1463.

R.L. Sani and P.M. Gresho (1994) Résumé and remarks on the open boundary condition minisymposium. *Int. J. Num. Meth. Fluids* 18, 983–1008.

A.E.P. Veldman (1990) Missing boundary conditions? Discretize first, substitute next, combine later. *SIAM J. Sci. Stat. Comput.* 11, 82–91.

R.W.C.P. Verstappen and A.E.P. Veldman (2003) Symmetry-preserving discretization for turbulent flow. *J. Comput. Phys.* 187, 343–368.

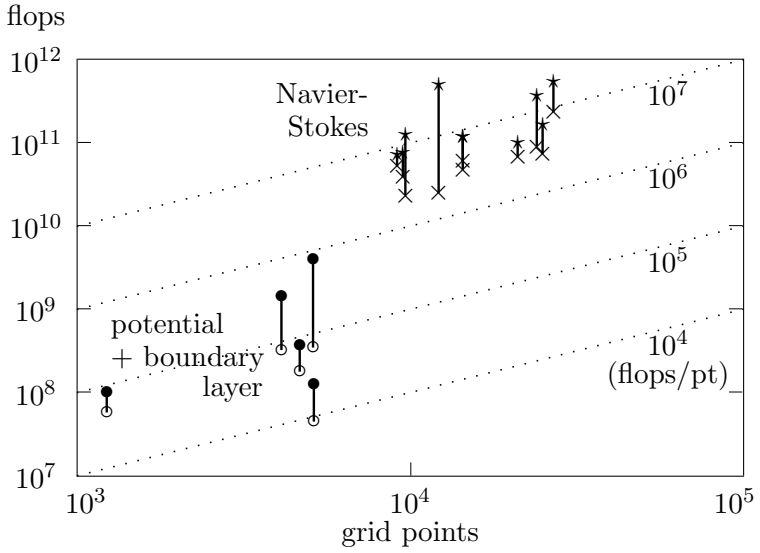
F.W. Wubs and J. Thies (2011) A robust two-level incomplete factorization for (Navier–)Stokes saddle point matrices. *SIAM J. Matrix Anal. Appl.* 32, 1475–1499.

Chapter 3

Direct numerical simulation of turbulence

3.1 Computational effort

Solving the Reynolds-averaged Navier–Stokes equations (RaNS) is a tremendous task, even for modern-day computers. A 1987 workshop on simulating viscous transonic flow around airfoils (2-D) shows that in practice between 10^6 and more than 10^7 floating point operations per grid point are required to obtain a steady solution (Holst 1987: see accompanying figure). For 3-D problems 10^7 flops/point are typical. A 3-D grid for an airplane e.g. contains $10^6 - 10^7$ grid points, with which a computation costs around 10^{14} floating point operations. It may be expected that in the future these numbers will decrease because of algorithmic improvements, such as the symmetry-preserving discretization described in Chapter 1. The figure



further shows that simulations based on boundary-layer methods are two orders cheaper than Reynolds-averaged Navier–Stokes. At present they do not provide less useful results than the Navier–Stokes methods, and therefore they are popular in the industrial design process.

The Reynolds-averaged Navier–Stokes equations model all turbulent phenomena. For certain applications this may not be accurate enough. The alternative is to resolve the large-scale turbulent phenomena (eddies) with sufficiently fine meshes and sufficiently small time steps. This is called *large-eddy simulation* (LES); see e.g. Reynolds (1990. Only the smallest structures are modelled; it is expected (or better: hoped) that they can be described in a more universal way. The required mesh widths are typically one order smaller than those for RaNS. Adding a corresponding smaller time step, a LES computation will be roughly four orders more expensive than a RaNS computation.

Most LES models make use of so-called *eddy diffusion* to dissipate the energy of the smallest flow structures that cannot be resolved by the grid. Experience over the years has shown that this not only removes energy from the small scales (as it should) but also from the larger scales in the flow. Herewith it influences too much the global flow pattern. In recent years, another approach, called *regularization modelling*, has been proposed in which the production of small scales through the convective terms is restricted[1]. This appears to be another step forward; see e.g. Trias *et al.* (2010, 2011).

If one wants to be relieved from finding an accurate turbulence model, than the grid should be sufficiently fine to resolve all flow scales in space and time, even the smallest ones. This is called *direct numerical simulation* (DNS) and comes with a corresponding computational price.

Estimates for the smallest scales in the flow have been made by Kolmogorov in 1941 for 3D homogeneous isotropic turbulence. He suggested that on the smallest flow scales an equilibrium exists between the production of these small scales and their destruction through viscous dissipation. Relevant quantities are the energy dissipation $\epsilon \equiv \mathrm{d}k/\mathrm{d}t$ ($\mathrm{m}^2/\mathrm{s}^3$), with k the turbulent kinetic energy per unit of mass, and the kinematic viscosity ν (m^2/s). Combining these quantities yields a unique length scale $\ell_K \sim (\nu^3/\epsilon)^{1/4}$ and time scale $\tau_K \sim (\nu/\epsilon)^{1/2}$. Thus the Kolmogorov length scale is proportional to $Re_{ed}^{-3/4}\,\ell_{ed}$, with a corresponding time scale proportional to $Re_{ed}^{-1/2}\,\ell_{ed}/u_{ed}$, in which ℓ_{ed} is the size of the large eddies and u_{ed} their velocity. Further, Re_{ed} is the Reynolds number based on the large eddies $Re_{ed} = u_{ed}\ell_{ed}/\nu$. This number is roughly one order smaller than the 'usual' Reynolds number (based on the global length scale), which for aerodynamic applications lies around 10^7, and for hydrodynamic applications around 10^9. For a 3-D calculation the grid would require $O(Re_{ed}^{9/4})$ points, with $O(Re_{ed}^{1/2})$ time steps. Hence, the complexity of a simulation scales with $Re_{ed}^{11/4}$! Thus a Reynolds number 10 times larger, requires almost a factor 1000 more computing effort.

Table 3.1 gives a (rough) indication of the computational power that is required to compute a DNS of flow past various objects. These estimates are based on the algorithms that have been developed in the last decade. It is humbling to compare these requirements with the performance of current (super)computers[2].

[1] In this way, the turbulence model does not interfere with the energy balance in the flow; as such these models fit in the energy-preserving philosophy already used in the discretization.

[2] Note that the flow of water is more expensive to compute than that of air: effectively air is more viscous than water!

The Dutch national supercomputer that was installed in 2000 aimed at an effective performance around 100 Gigaflop/s (peak 1 Teraflop/s)[3] Starting from here, about five orders of magnitude have to be bridged to perform a DNS of a turbulent flow at $Re = 10^7$. Assuming that both computer hardware and computational algorithms will continue to progress at the rate that they have developed during the past three decades – both have become one and a half order of magnitude faster per decade – within two decades the lacking five orders of magnitude could be bridged. For this estimate to come true, computers need to become about a thousand times faster and must be supplied with one thousand times as much memory within the next two decades. Within that span of time the numerical algorithms for DNS need to become three orders of magnitude faster, need to run efficiently at the fastest available machines, and need to use three orders of magnitude less memory than today's algorithms do require.

	medium	*speed*	*Reynolds*	*grid points*	*performance (2 week run)*
cyclist (tourist)	air	20 km/h	$1 \cdot 10^5$		
golf ball (pro)	air	250 km/h	$2 \cdot 10^5$	10^9	$10^{11.5}$ flop/s
speed skater (pro)	air	45 km/h	$5 \cdot 10^5$		
swimmer (pro)	water	5 km/h	$3 \cdot 10^6$	10^{11}	10^{14} flop/s
car	air	80 km/h	$5 \cdot 10^6$		
shark	water	20 km/h	$2 \cdot 10^7$	10^{13}	$10^{16.5}$ flop/s
airplane	air	900 km/h	$3 \cdot 10^7$		
ship	water	20 km/h	up to 10^9	10^{17}	10^{21} flop/s

Table 3.1: *Overview of (estimated) required computational resources (CPU performance) for DNS at Reynolds numbers in the range* $10^5 - 10^9$ *(2010 algorithms).*

In fact, the ASCI project in the USA, initiated in the mid 1990-ies, strived for this type of performance improvement. In 1999 already three computers existed where the 1 Teraflop/s limit has been achieved. The Supercomputer TOP500 of November 1999 was led by the products of the ASCI project: on top of the list Intel's ASCI Red with 2.4 Teraflop/s, followed by IBM's Blue Pacific with 2.1 Teraflop/s and SGI's Blue Mountain with 1.6 Teraflop/s. However, the type of application for which these processor speeds are achieved, the so-called Linpack benchmark, is not fully representative for CFD. Linpack is very cache-friendly, i.e. once data have arrived in cache the algorithm makes frequent of use of them before they are replaced by fresh data. In CFD algorithms the communication with main memory is much more frequent, and a factor of three (or more) is easily lost compared to Linpack performance. Thus, for CFD simulations computers with high access rates of memory (high bandwidth/low latency), such as vector computers where up 30-50% of peak performance can be achieved in practice, are very much in favour over commodity processors like Intel's Xeon or AMD's Opteron. On the latter machines only around 5% of peak performance can be achieved for CFD algorithms.

The TOP500 list has been dominated by (North-American) massively parallel cache-based (scalar) computers, under a constant challenge of (Japanese) computers based on vector ar-

[3]Giga= 10^9, Tera= 10^{12}, Peta= 10^{15}, Exa= 10^{18}, Zetta= 10^{21}, Yotta= 10^{24}.

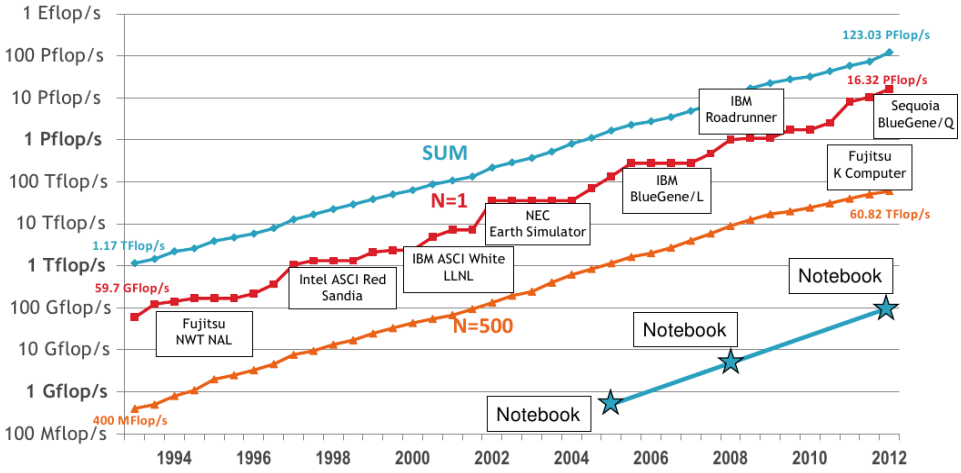


Figure 3.1: *Development of computer performance as illustrated by the Top500 list: top entry, #500 and the sum of all entries (Meuer and Gietl 2012).*

chitecture; Figure 3.1 shows the developments between 1993 and 2012. E.g. for about three years, 2002-2004, the fastest computer was the Japanese NEC Earth Simulator, based on vector processors, where 12 Teraflop/s sustained(!) performance could be achieved on codes for climate modelling (36 Teraflop/s on Linpack). In June 2009, when the TOP500 was headed by several Intel, IBM and AMD based clusters, the two toppositions broke the Petaflop/s (= 1000 Teraflop/s) barrier. Since then, the progress in cluster performance has continued. In June 2012 the topposition, an IBM Blue Gene/Q, reached over 16 Petaflop/s on a cluster with 1.6 million(!) cores. Plans are being made to attack the Exaflop/s barrier (10^{18} flop/s), hopefully to be reached in 2019 (Meuer and Gietl 2012).

Thus, the progress in computer hardware remains impressive. Together with the equally impressive progress in numerical algorithms, DNS computations at full-scale Reynolds numbers glimmer at the horizon.

3.2 Navier–Stokes: higher-order space discretization

It will be clear that DNS requires the utmost of computers and algorithms. Therefore much research is going on in improving the efficiency of algorithms. At RUG we have concentrated on higher-order discretization methods along the lines presented in Section 1.5. We will apply the above ideas now to construct a higher-order discretization of the Navier–Stokes equations on the staggered grid as discussed above. In the described second-order method the x-momentum equation is applied to a control volume as shown in Figure 3.2 (left). We note that the vertical velocity is defined in the corner points of this volume.

The control volume now is combined with a larger one: for the convection-diffusion equation above we chose a two-times larger volume, but here we will use a three-times larger volume as indicated in Figure 3.2 (right). The latter choice is the smallest volume for which the corners coincide with positions of the vertical velocity. The coinciding of corners with ver-

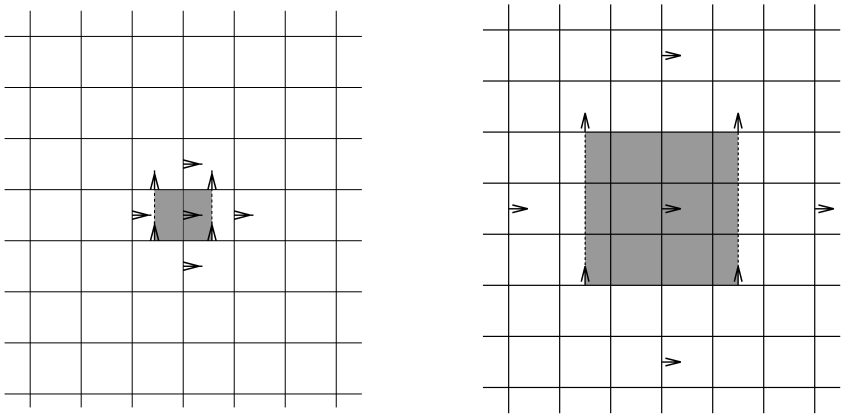


Figure 3.2: *The left picture shows a control volume for the conservation of the horizontal component of the momentum field (in two spatial dimensions). The right picture shows a three-times larger control volume that is applied to eliminate the leading term of the truncation error. The arrows denote the components of the discrete velocity that are used to discretize the application of the conservation law to the control volume.*

tical velocities implies that one discretization method can be used for the spatial integration of the momentum equation over both the original control volumes and the larger volumes. Two times larger control volumes, for instance, have corners that do not coincide with positions of the vertical velocity, and thus do require additional high-order interpolations (e.g. for the cross terms in the convective flux). This interpolation unavoidably introduces some diffusion in the discretization, which does interfere with the real diffusion as we have experienced in test calculations. For DNS this is an unacceptable situation.

As in the convection-diffusion example above, near the boundaries we simply follow the second-order approach to avoid problems with control volumes that do not fit in the domain.

We update the velocity by integrating the momentum equations over one time step and by correcting the result by adding the pressure gradient in such a way that the mass is conserved in control volumes centred on the pressures. Obviously, we can also apply the balance of mass to three times larger control volumes and use these control volumes to eliminate the leading term in the truncation error. Thus, a fourth-order-accurate spatial discretization can be derived for the continuity equation. Since the 'div' and 'grad' operator analytically are each other's transpose (apart from a minus-sign; see Section 2.3), the discretization of the divergence immediately implies the discretization of the gradient. Subsequently, the discrete Laplacian (= div grad) appearing in the pressure Poisson equation is implied. More elaborated details can be found in Verstappen and Veldman (2003).

The discrete Poisson equation for the pressure is solved by means of the conjugate gradient method with modified incomplete Choleski preconditioning according to Gustafson (1978). The MICCG code is fully vectorized by an explicit reordering of the unknowns along diagonals of grid planes parallel to the symmetry plane of the cavity. The implementation of the preconditioned iterative method is optimized as proposed by Eisenstat (1981). The initial guess for the iteration is obtained by extrapolation of the pressure from previous time levels.

2nd- vs. 4th-order discretization

As an example of the progress made, we present a comparison between a second-order approach and our fourth-order approach for flow in a 3D driven cavity at $Re = 10,000$. Experimental results are available for comparison (Prasad and Koseff 1989). We show the vertical velocity along the central axis of the cavity as obtained from a second-order calculation on a $100 \times 100 \times 100$ grid, a fourth-order calculation on a $50 \times 50 \times 50$ grid and the experimental results. The overall error in an instantaneously measured velocity is about $\pm$ 0.6% of the maximum velocity. The stretching in the latter grid is moderate: the grid spacing is exponentially stretched away from the wall, and the largest mesh width is approximately seven times wider than the smallest.

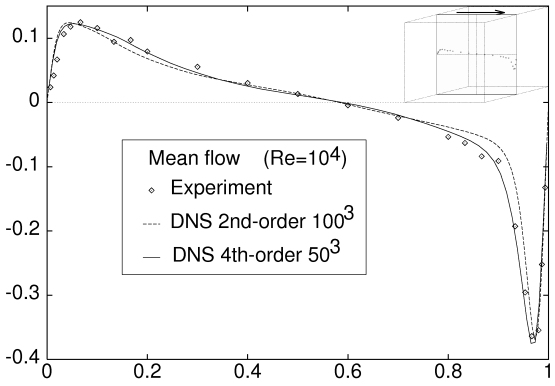


Figure 3.3: *Lower-order and higher-order computation vs. experiment of turbulent flow in a 3D driven cavity at* $Re = 10,000$. *Shown is the normal velocity along the central axes of the symmetry plane.*

In this example the fourth-order results are clearly superior to the second-order results, whereas the computational effort is about twenty times less. Indeed, the CPU-time per grid point and time step of both methods is comparable, a 50^3 grid has 8 times less grid points than a 100^3 grid, it allows for a twice as large a time step (note that the time step is restricted by the CFL-condition), and the number of iterations needed to solve the pressure correction from the Poisson equation is somewhat less for a 50^3 grid than for a 100^3 grid: $8 \times 2 \times 1.25 = 20$. Essential to this success is the way in which the discretization of the convective terms copes with the stretching of the computational grid.

3.3 Refined time integration

In these lecture notes, mostly the explicit Forward-Euler time-integration method has been used; not because it is the most efficient method, but it suits didactical purposes best. Runge–Kutta methods and multi-step methods (like Adams–Bashforth) – to stay within the explicit methods – provide more efficiency. In this section we will discuss a refinement of the latter, which is better suited for (turbulent) flow simulations at large Reynolds numbers (Verstappen and Veldman 1998, 2003).

In Section 1.8 it was concluded that the diffusive stability limit is most restrictive when the cell-Péclet number $P \equiv uh/\nu < 2$, but for $P > 2$ the convective limit dominates. Close to a solid wall the flow is laminar, with low velocities due to the (no-slip) boundary condition. Here, in wall-normal direction we have locally $P < 2$ and the diffusive limit is the most restrictive. Further away from the wall the flow is turbulent and Kolmogorov's estimates apply.

For a grid size proportional to the Kolmogorov length scale $\ell_K \propto \nu^{3/4}$, the cell-Péclet number scales with $\nu^{-1/4}$ and is likely to be > 2 (which should not be too bad as the resolution is OK). The (mathematical) convective stability limit becomes $\delta t_{\rm conv} \propto \nu^{3/4}$, much less than the (physical) Kolmogorov time scale $\tau_K \propto \nu^{1/2}$. Thus, in the bulk of the flow domain the convective stability limit is our main concern. Therefore, below we will present a simple adaptation of the Adams–Bashforth method which is tuned to this situation and enhances its efficiency by a factor of two.

Hereto we consider a family of explicit second-order one-leg methods, which will be discussed with aid of the test problem $u' = f(u)$. For this problem the one-leg family reads

$$(\alpha + \tfrac{1}{2})u^{(n+1)} - 2\alpha u^{(n)} + (\alpha - \tfrac{1}{2})u^{(n-1)} = \delta t f\left((1+\alpha)u^{(n)} - \alpha u^{(n-1)}\right). \qquad (3.1)$$

This discretization is second-order accurate for all $\alpha \neq -\frac{1}{3}$, and third-order accurate when $\alpha = -\frac{1}{3}$. Its error constant is given by $C_3 = \frac{1}{6}(1+3\alpha)$. The 'one' in one-leg refers to the fact that these methods evaluate the right-hand side f at one point only. For more details on one-leg methods the reader is referred, for instance, to Hairer (1991).

Taking $\alpha = \frac{1}{2}$, a one-leg method is created which is the twin of Adams–Bashforth. According to Adams–Bashforth we ought to take $\frac{3}{2}f(u^{(n)}) - \frac{1}{2}f(u^{(n-1)})$ instead of $f(\frac{3}{2}u^{(n)} - \frac{1}{2}u^{(n-1)})$. One- and two-leg methods are identical when f is linear with time-independent coefficients, and then they have the same region of linear stability. They differ when the right-hand side f is non-linear or explicitly time-dependent. For instance, in the 'simple' situation where $f(u) = \lambda(t)$ with $\lambda(t) < 0$ (i.e. u does not enter the right-hand side), the solution $u^{(n+1)}$ of (3.1), i.e.

$$u^{(n+1)} = u^{(n)} + \delta t\, \lambda(\tfrac{3}{2}t_n - \tfrac{1}{2}t_{n-1}),$$

is smaller than $u^{(n)}$ (for any time step) as it should be. On the other hand, the solution obtained with the associated multi-step method given by

$$u^{(n+1)} = u^{(n)} + \delta t\, \{\tfrac{3}{2}\lambda(t_n) - \tfrac{1}{2}\lambda(t_{n-1})\}$$

does not satisfy the inequality $u^{(n+1)} < u^{(n)}$ unconditionally (it depends on the variation of λ with time). In addition, it is emphasized in Nevanlinna (1978) that (some) one-leg methods are more reliable than their corresponding multi-step methods when used with variable time steps.

For $\alpha = 0$ the leapfrog method is obtained. This method cannot be used to integrate a diffusive flux in time, since it is not stable: the linear stability region of leapfrog consists of all purely imaginary numbers with modulus smaller than or equal to one.

Our aim is to determine α such that the corresponding method allows for the largest time step, or stated otherwise, possesses the largest region of convective stability. Figure 3.4 (left) shows the stability domain of the one-leg method for $\alpha = 0.05$ and $\alpha = 0.5$ (Adams–Bashforth). The stability domain is pressed against the imaginary axis when α goes to zero. In the limit $\alpha = 0$ the stability domain is equal to the interval $[-i, i]$. Above we concluded that convective stability puts the most severe restriction on the time step. Thus, we look for stability domains which include eigenvalues $\lambda = x + iy$, where the real part x is negative and

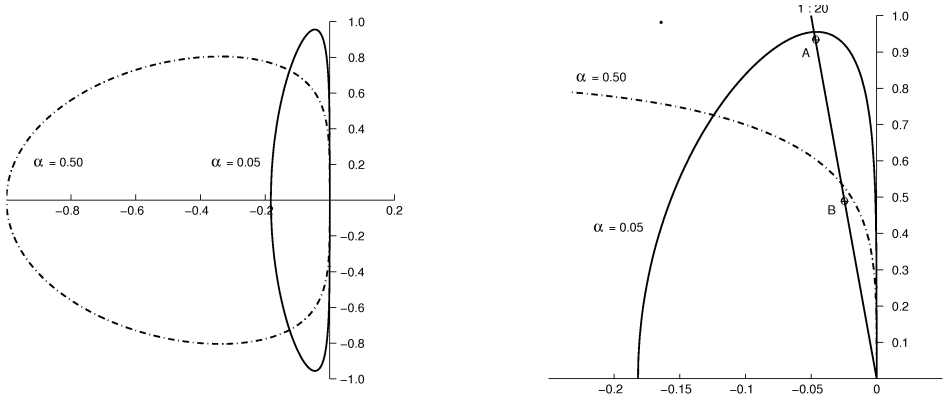


Figure 3.4: *The left picture shows the stability domain of the one-leg method (3.1) for* $\alpha = 0.05$ *and* $\alpha = 0.5$. *The right picture shows a blow up of the stability domains near the positive imaginary axis.*

the absolute value of the imaginary part y is much larger than the absolute value of the real part. Here, 'much' can range from one to two orders of magnitude. For a DNS of a flow in a driven cavity at $Re = 10^4$ with a grid size $\delta x = 10^{-2}$, and a maximum velocity $U_{\max} = 1$, for example, $|x| : |y|$ is of the order of $1 : 100$.

Under these conditions, the one-leg method with $\alpha = 0.05$ outperforms Adams–Bashforth. Figure 3.4 (right) shows a blow up of the stability domains of both methods near the positive imaginary axis. The points denoted by A and B lie on the line $|x| : |y| = 1 : 20$. The point A lies close to the boundary of the stability domain for $\alpha = 0.05$; B lies near the boundary of the stability domain for $\alpha = 0.5$. A lies approximately two times as far from the origin as B. Thus, the time step of the one-leg method with $\alpha = 0.05$ can be enlarged by a factor of two compared to Adams–Bashforth. For $|x| : |y| = 1 : 10$ this factor is about 1.5; for $|x| : |y| = 1 : 100$ it is approximately 2.5. We have carried out a number of numerical test calculations of the (2D) flow in a driven cavity at $Re = 10^3 - 10^5$. The results demonstrate that the one-leg method with $\alpha = 0.05$ requires indeed about two times less computational effort than the Adams–Bashforth method, whereas the accuracy is just as good.

3.4 Examples of turbulent-flow simulation

Both the algorithmic improvements and the improvements in computer performance have opened the door to solve the Navier-Stokes equations numerically for Reynolds numbers in the range $10^4 - 10^5$ without using any turbulence model. Results in this range will first be shown in comparison with LES concerning the flow past a long, square cylinder at Re = 22,000 (Section 3.4.1). Thereafter, in Section 3.4.2 a comparison is made between DNS and RaNS for a flow past an array of cubes. Finally (Section 3.4.3), results for a channel flow are presented for which experimental results and early DNS computations are available. For more details see Verstappen and Veldman (1996, 1997, 1998).

3.4.1 Flow past a square cylinder at Re = 22,000

The comparison with LES concerns the flow past a long, square cylinder at Re = 22,000 (at zero angle of attack). We assume that the flow is periodical in the spanwise direction. The

spanwise boundaries are taken four diameters apart. We prescribe laminar inflow at six and a half diameters upstream of the cylinder. Experiments by Lyn *et al.* (1995) indicate a turbulence level (*i.e.* the ratio between the fluctuating and the mean velocity) of about 2% at four and a half diameters upstream from the cylinder. The lateral boundaries are taken 14 diameters apart. The outflow boundary is positioned at 20 diameters past the cylinder. In addition, in a buffer zone (of five diameters length) the Reynolds number is decreased from 22,000 to 1,000 to suppress (non-physical) waves which are reflected by the artificial outflow boundary. The (artificial) outflow conditions read $v_{nn} = w_{nn} = 0$ and $p_n =$ constant, where the constant is determined such that the mass inflow equals the mass outflow (at each time-step); this constant is approximately zero. No-slip boundaries are imposed at the surface of the cylinder.

We have used a $280 \times 210 \times 64$ staggered grid to cover the computational domain. The first grid point is spaced 0·005 from the cylinder surface. The grid is stretched out away from the cylinder surface by means of a sinh function; the ratio of the largest to smallest grid size in the streamwise direction is approximately 200. About 5% of the grid points are located in the buffer zone. The discrete Poisson equation for the pressure is solved with a combination of a Fast Fourier Transform method in the spanwise direction and a modified incomplete Choleski Conjugate-Gradient method in the resulting spectral space.

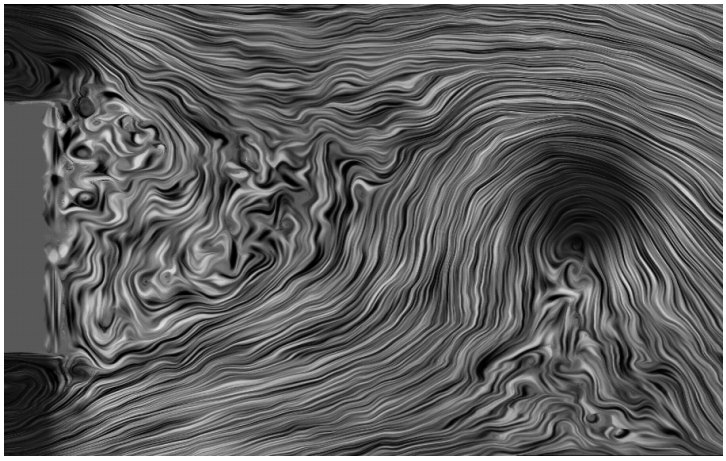


Figure 3.5: *Snapshot of the flow past a rectangular cylinder at Re=22,000. The visualisation has been done with the spot-noise method (De Leeuw 1997).*

The start-up of the flow plus three shedding cycles have been computed on 16 nodes of a CRAY J932. We compute averages over three shedding cycles by sampling the flow at each time step. Velocities are also averaged over the top and bottom halves. All quantities are normalized by the cylinder width and the inflow velocity.

	Simulation	*Experiments*		
		Lyn *et al.* (1995)	Lee (1975)	Vickery (1966)
Mean Strouhal number	0·133	$0{\cdot}133 \pm 0{\cdot}003$	–	–
Mean drag coefficient C_d	2·09	2·1	2·05	2·05
Mean lift coefficient C_l	0·005	–	–	–
Rms fluctuation of C_d	0·178	–	$0{\cdot}16 - 0{\cdot}23$	–
Rms fluctuation of C_l	1·45	–	–	$0{\cdot}68 - 1{\cdot}32$

Table 3.2: *Comparison with experiment: bulk quantities*

Table 3.2 shows the mean Strouhal number, the mean drag coefficient C_d, the mean lift coefficient C_l and the root-mean-square fluctuations of C_d and C_l. Here, it may be noted that the mean lift coefficient has not been measured; it should be zero by symmetry. All computed bulk quantities fall within the range set by the experiments, except for the root-mean-square of the fluctuations of the lift coefficient C_l which is slightly overestimated. Figure 3.6 shows a comparison of mean streamwise velocities with experimental data at four locations past the cylinder. The agreement between the experimental data and the numerical data is good.

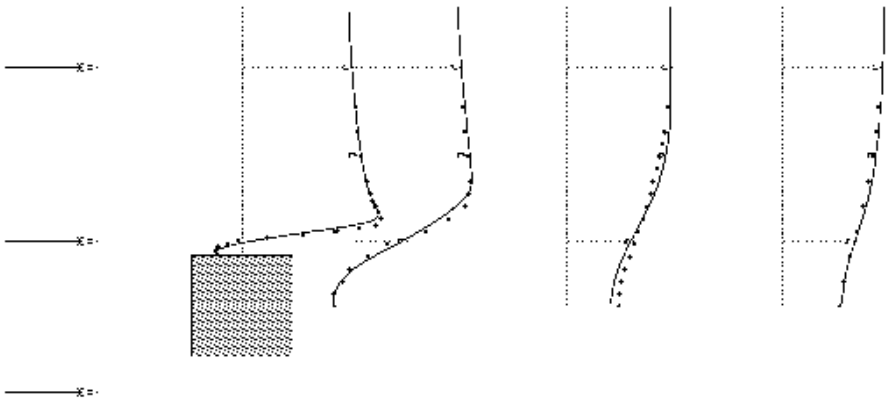


Figure 3.6: *Flow past a square cylinder at Re=22000: a comparison of mean velocities of the fourth-order simulation with experimental results. The experimental data is taken from ERCOFTAC Database Case 43; see also Lyn et al. (1995). Shown is the mean streamwise velocity. The continuous lines correspond to the simulation; the experimental data is depicted by the dots.*

The flow past a square cylinder at Re = 22,000 (at zero angle of attack) has served as a test case at two LES workshops. The results submitted to the workshops can be found in Rodi (1997) and Voke (1997). The results show a great variation in the predictions, and it was concluded that this flow forms a major challenge to current LES techniques; no one has found the definite solution, yet. There is no doubt that the definite solution has to be computed by means of an accurate, cost-effective numerical method. In Figure 3.7, we have plotted the drag coefficients of almost all the submissions versus the numerical method that has been applied. Here, we have left out a few, namely those that had serious failings (a too large lift coefficient, or a shedding frequency that differed significantly from the experiments).

There are of course many factors that (can) affect the prediction of the drag coefficient: the subgrid scale model, the value of parameters in the model, the grid resolution, the numerical approach, the treatment of the walls, etc. Unfortunately, these factors varied during the workshops, and consequently it is difficult to trace the source of the errors in the results of the simulations.

As can be seen in Figure 3.7, large-eddy simulations with a QUICK or a third-order upwind (U3) discretization predict a drag force which is too large. The U3-approach that comes closest to the experimental data uses approximately five times the number of grid points of each of the other QUICK/U3-methods. Although we cannot draw firm conclusions, for the

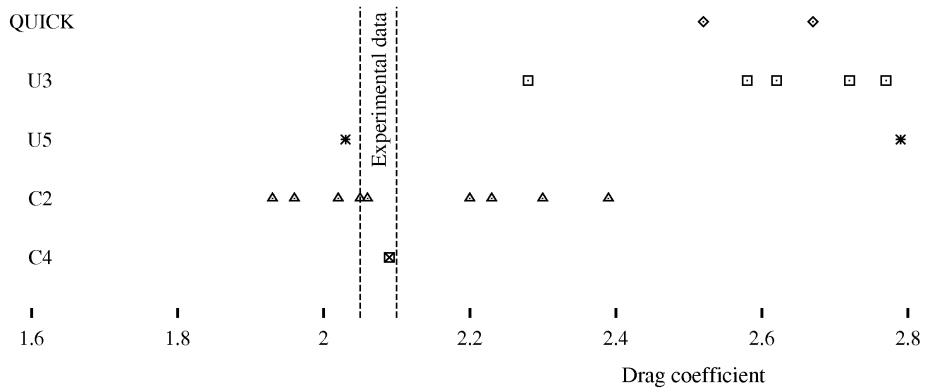


Figure 3.7: *Flow past a square cylinder at Re=22,000: the mean drag coefficient versus the discretization of the convective terms of the Navier–Stokes equations. Here, U stands for upwind, and C for a central discretization. The number denotes the order of the method: C2, for example, represents a second-order central discretization. The experimental results of Lyn et al. (1995), Lee (1975) and Vickery (1966) fall within the shaded area. The results of the large-eddy simulations are taken from Rodi (1997) and Voke (1997).*

reason mentioned above, it is most likely that the artificial diffusion added by QUICK/U3 contaminates the results.

The performance of subgrid models is difficult to judge since modelling errors and numerical errors interact; see also Vreman *et al.* (1994). To separate these errors, a number of subgrid models need to be considered within the same numerical approach. We feel that symmetry-preserving methods provide a good basis for evaluating turbulence models, since they are accurate on non-uniform grids and the character of the numerical error is well-defined: the error in the convective term is convective; the error in the diffusive term is diffusive.

3.4.2 Surface mounted cubes

Next we show an example of a turbulent flow that has served as a test case for turbulence modelling at the 6th ERCOFTAC/IAHR/COST Workshop on Refined Flow Modelling which was held at Delft University of Technology in June 1997 (Hanjalić and Obi 1997). The problem considered is the fully developed flow in a channel where an array of 25×10 cubes is placed regularly at the bottom. The Reynolds number based on the height of the channel and the bulk velocity is 13,000. The channel is 3.4 times wider than a cube. The pitch of the cubes is four cube lengths in both the streamwise and the spanwise direction.

Meinders *et al.* (1998) have measured this complex turbulent flow. Their flow measurements around the 18th row from the inlet showed that the influence of the in- and outlet can be neglected there. Hence, the computational domain for a numerical simulation can be confined to a sub-channel unit of dimension $4h \times 3.4h \times 4h$ (where h denotes the height of a cube) with periodic boundary conditions in both the streamwise and the spanwise directions. Figure 3.8 displays the sub-channel unit.

We have applied the fourth-order, symmetry-preserving method that is outlined in Section 3.2 to compute mean velocity profiles and Reynolds stresses at various locations in the

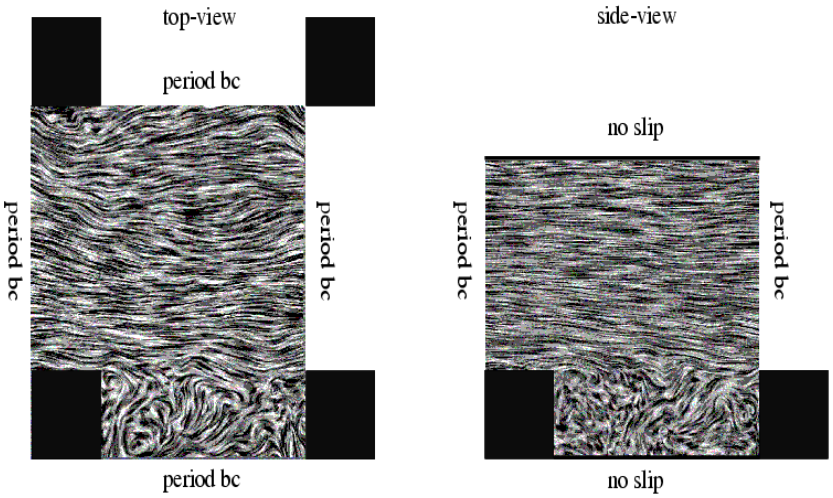


Figure 3.8: *Top- and side-view of a sub-channel unit. An instantaneous flow field at two planes through the centre of the cubes is shown, as obtained from the* 100^3 *DNS. In both pictures the flow is directed from left to right. It may be noted that large structures of recirculating flow behind the obstacles are not present in any of the snapshots of this flow. These regions can only be observed if the flow is averaged over a long period in time.*

sub-channel (Verstappen and Van de Velde, 2006). To this end, the sub-channel is covered by a 100^3 grid. The first mesh point is spaced 0·006 to a wall. A cube is represented by 40 grid points in each direction. The grid is slightly stretched: the largest grid size is approximately three times the smallest. No-slip boundaries are imposed at the surfaces of the cubical obstacles and also on the walls of the channel.

The entire computation (including start-up and sampling time) took about 100 hours on one vector-processor of a CRAY C90. Mean velocity profiles as well as Reynolds stresses at various locations in the channel have been computed. The sampling of data for the computation of the first- and second-order statistics of the flow started after a transitional period of 100 (non-dimensional) time units. The time-averages were computed over 200 time units and over the two symmetrical halves of the sub-channel unit. Samples were taken at each time step.

Figure 3.9 shows a comparison of the first- and second-order statistics[4] of the 100^3 simulation with the experimental data of Meinders *et al.* (1997) in the symmetry plane parallel to the streamwise direction that bisects a cube. The agreement between the experimental data and the numerical data is excellent.

In order to challenge RaNS computations, we have also performed a simulation at a 60^3 grid. On this grid, a cube is represented by 30 grid points in each direction. The first grid point of the 60^3 grid lies approximately twice as far from the wall than the first grid point of the 100^3 grid. The coarse grid simulation takes about one tenth of the CPU-time of the

[4]Because of the seemingly chaotic behaviour of turbulent flow, it makes little sense to compare snapshots in time as in Figure 3.8 directly. Instead, one compares time averages and fluctuations around the mean as in Figure 3.9.

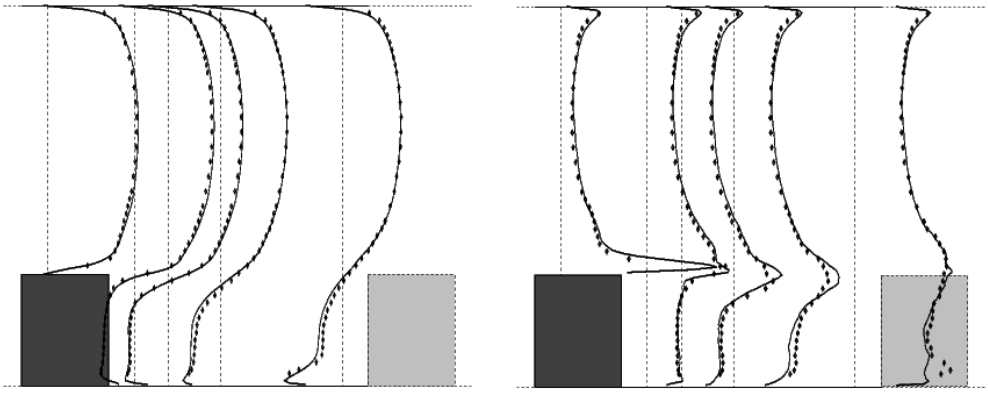


Figure 3.9: *Comparison of numerical results (continuous lines) with experiment (dots) for flow past an array of cubes. The left picture shows the mean streamwise velocity $\bar{u}$, whereas the right picture shows the fluctuating streamwise velocity $\overline{u'u'}$.*

fine grid simulation, among other things because the time step can be increased by a factor of about two.

There were four groups who presented results of their RaNS computations of the flow in the channel with surface-mounted cubes at the workshop (Hanjalić and Obi 1997). Here, we restrict ourselves to the RaNS computation (of Dr. S. Jansson from the Department of Fluid Mechanics of Vattenfall Entvecklung AB in Sweden) that agreed the best with the available experimental data. It may be noted that there were no results of large-eddy simulations submitted to this workshop, nor have there been any reported elsewhere.

The best RaNS result was based on a second-order, cell-centered, finite-volume method. The QUICK scheme was used for the velocities and the turbulent quantities were integrated by means of a second-order accurate scheme with a Van Leer limiter. Periodic boundary conditions were applied in the streamwise direction; the period was taken equal to four cube lengths. Symmetry conditions were applied in the spanwise directions. The spanwise boundaries were taken two cube lengths apart. Dirichlet conditions were applied at the solid walls, except for the dissipation rate ϵ: the normal derivative of ϵ was put to zero at solid walls. The turbulence model consisted of a two-layer eddy-viscosity combined with a standard $k-\epsilon$ model. The transport equation for the dissipation rate ϵ was not solved in near-wall regions, but instead it was computed explicitly from a predicted length scale. This RaNS computation was performed on a stretched, orthogonal grid of $67 \times 72 \times 57$ points in the streamwise, the normal and the spanwise direction, respectively. Except for the spanwise direction, the grid spacing of this RaNS computation is slightly finer than that of our 60^3 simulation. In the spanwise direction, the average resolution of the RANS computation is about two times finer than the average resolution of the 60^3 grid, due to the fact that the RaNS computation uses symmetry conditions and therefore can restrict its spanwise computational domain to two cube lengths, while in our simulation periodic boundary conditions have been applied in the spanwise direction with a period of four cube lengths.

Mean streamwise velocities are compared in Figure 3.10. On a corresponding grid, the mean velocities computed from the Reynolds averaged Navier–Stokes equations agree less

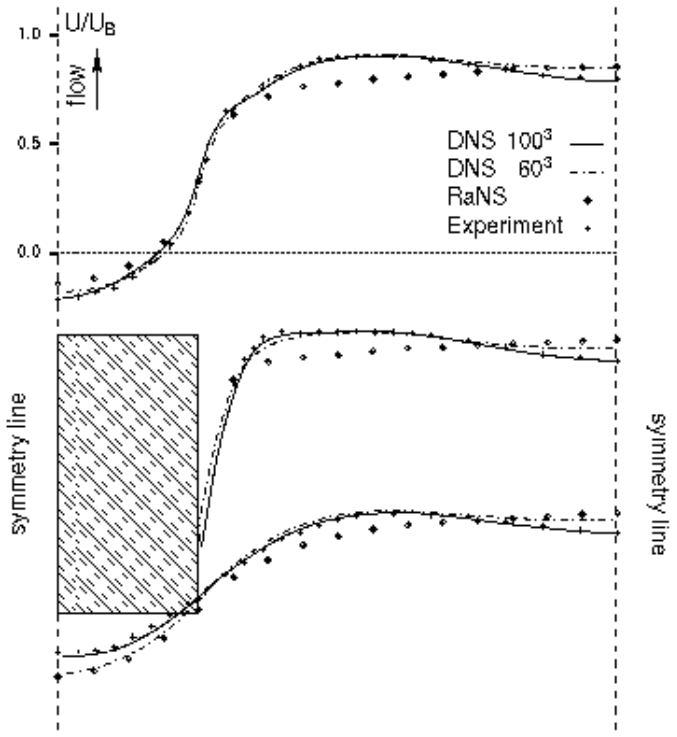


Figure 3.10: *Comparison of the mean streamwise velocity at half cube height. The flow comes from below; the horizontal corresponds to the spanwise direction. The dashed vertical lines are lines of symmetry; their distance is two cube lengths. The lowermost profiles are located at 0.3 cube lengths before the front of the cube, the middlemost profiles at 0.3 cube lengths after the front an the uppermost profiles at one cube length after the middlemost. The velocity scale is shown for the uppermost profiles only. The experimental data is taken from Meinders (1997); the RaNS data is taken from Hanjalić and Obi (1997).*

with the experimental data than the results of the symmetry-preserving simulation. The velocity profiles of the RaNS computation are much too smooth. In addition, the maxima of the velocities are located in the symmetry-plane between two cubes, which is in distinct disagreement with the experimental data.

Finally, it may be observed that the convergence of the symmetry-preserving simulation upon grid refinement is plain: the results on the 100^3 grid are closer to the measurements than those of the 60^3 grid.

3.4.3 Channel flow

To further illustrate the accuracy of the symmetry-preserving discretization, consider a turbulent channel flow at a Reynolds number of Re=5,600. A large number of numerical results as well as experimental data is available for comparison. Figure 3.11 (left) shows a comparison of the mean velocity profile as obtained from the 4th-order symmetry-preserving simulation with those of other direct numerical simulations. The grids used by the DNS's that we compare with have typically about 128^3 grid points, that is 16 times more grid points than our grid has. Nevertheless, the agreement is excellent.

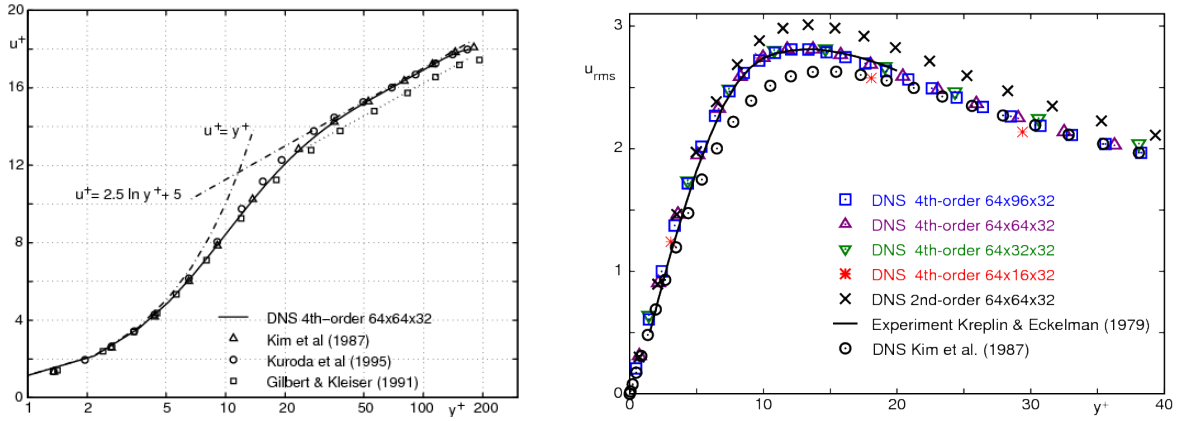


Figure 3.11: *Comparison of the mean streamwise velocity u^+ (left) and its root-mean-square (right) as function of the wall normal coordinate y^+.*

The root-mean-square of the fluctuating streamwise velocity u_{rms} near the wall ($0 < y^+ < 40$) is presented in Figure 3.11 (right). The results show that the u_{rms} predicted by the 4th-order simulation fits the experiment data of Kreplin and Eckelmann (1979) nicely. This holds for very coarse grids too, as can be inferred from the results computed with the 4th-order symmetry-preserving scheme with only 32 grid points in the wall-normal direction. It is interesting to compare the ratio between 'our' 32 or 64 grid points in wall-normal direction and the 128 grid points used in the reference computations (Kim *et al.* 1987, Kuroda *el al.* 1995, Gilbert and Kleiser 1991). This ratio (a factor two to four) compares quite well with the 'predicted' possible increase in grid size (a factor of three) as found in the one-dimensional test case described in Section 1.7.

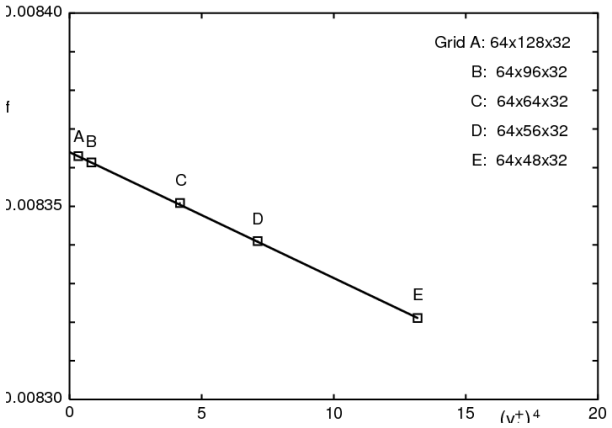


Figure 3.12: *Convergence of the skin-friction coefficient upon grid refinement, displayed as a function of the fourth power of the cell size next to the wall.*

This flow case has also been used to carry out a grid refinement study to show indeed 4th-order convergence behaviour. The skin-friction coefficient C_f has been monitored as obtained from simulations on five different grids, denoted A – E. Their spacings differ only in the direction normal to the wall, with 128 (A), 96 (B), 64 (C), 56 (D) and 48 (E) points,

respectively. The first (counted from the wall) grid line varies between $y_1^+ = 0.72$ (grid A) and $y_1^+ = 1.9$ (grid E). The refinement study, as indicated in Figure 3.12, clearly shows a 4th-order behaviour on the non-uniform grid. The straight line in Figure 3.12 is approximately given by $C_f = 0.008364 - 0.000004(y_1^+)^4$. The extrapolated value at the crossing with the vertical axis $y_1^+ = 0$ lies in between the $C_f = 0.00818$ reported by Kim *et al.* (1987) and Dean's correlation $C_f = 0.073Re^{-1/4} = 0.00844$. Note that the extrapolation eliminates the (leading term of the) discretization error in the wall-normal direction, but not the other discretization errors in space and time.

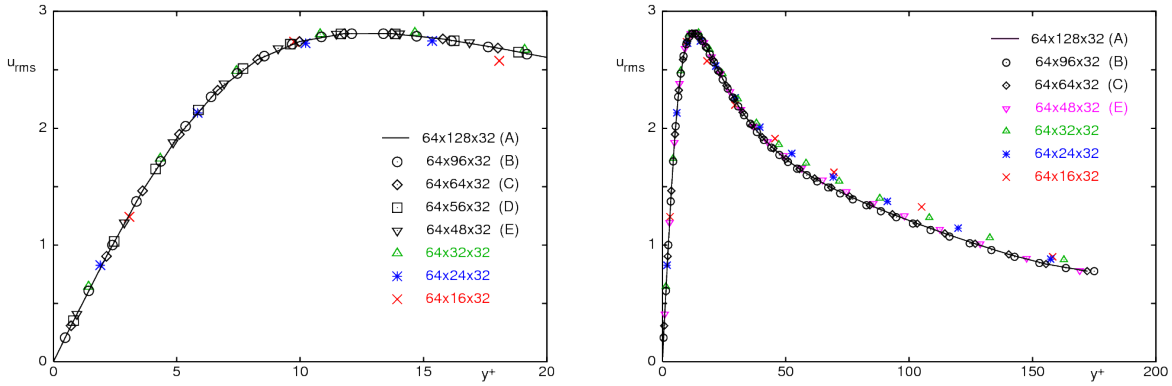


Figure 3.13: *The root-mean-square velocity fluctuations normalized by the wall shear velocity as a function of the wall coordinate y^+ on various grids for $y^+ \leq 20$ (left) and $y^+ \leq 200$ (right).*

The convergence of the fluctuating streamwise velocity near the wall ($0 < y^+ < 20$) is presented in Figure 3.13 (left). Here, we have added results obtained on three still coarser grids (with 32, 24 and 16 points in the wall-normal direction, respectively), since the results on the grids A–E fall almost on top of each other. The coarsest grid, with only 16 points to cover the channel width is coarser than most of the grids used to perform a large-eddy simulation (LES) of this turbulent flow. Nevertheless, in the near wall region, the $64 \times 16 \times 32$ solution is not that far off the solution on finer grids. Further away from the wall, the turbulent fluctuations predicted on the coarse grids (32 points or less in vertical direction) become too high compared to the fine grid solutions, as is shown in Figure 3.13 (right).

3.5 References

S.C. Eisenstat (1981) Efficient implementation of a class of preconditioned Conjugate Gradient methods, *SIAM J. Sci. Stat. Comput.* 2, 1–4.

N. Gilbert and L. Kleiser (1991) Turbulence model testing with the aid of direct numerical simulation results. In: *Proc. Turb. Shear Flows 8*, paper 26-1.

I. Gustafson (1978) A class of first order factorization methods. *BIT* 18, 142–156.

E. Hairer and G. Wanner (1991) *Solving Ordinary Differential Equations II.* Berlin: Springer-Verlag.

K. Hanjalić and S. Obi (eds.) (1997) *Proc. 6th ERCOFTAC/IAHR/COST Workshop on Refined Flow Modeling*, Vol. 3, Section Heat Transfer, Delft University of Technology, 38 pages.

T.L. Holst (1987) Viscous transonic airfoil workshop compendium of results. *AIAA paper* 87–1460.

J. Kim, P. Moin and R. Moser (1987) Turbulence statistics in fully developed channel flow at low Reynolds number. *J. Fluid Mech.* 177, 133–166.

L. Kleiser and T.A. Zang (1991) Numerical simulation of transition in wall-bounded shear flows. *Annu. Rev. Fluid Mech.* 23, 495–537.

H.P. Kreplin and H. Eckelmann (1979) Behavior of the three fluctuating velocity components in the wall region of a turbulent channel flow. *Phys. Fluids* 22, 1233–1239.

A. Kuroda, N. Kasagi and M. Hirata (1995) Direct numerical simulation of turbulent plane Couette-Poiseuille flows: effect of mean shear rate on the near-wall turbulence structures. In: F. Durst *et al.* (eds.) *Proc. Turb. Shear Flows 9.* Berlin: Springer-Verlag, pp. 241–257.

B.E. Lee (1975) The effect of turbulence on the surface pressure field of square prisms. *J. Fluid Mech.* 69, 263–282.

W. de Leeuw (1997) *Presentation and Exploration of Flow Data.* Ph.D. Thesis, Delft University of Technology, Delft, 107 pages.

D.A. Lyn, S. Einav, W. Rodi and J.H. Park (1995) A laser-Doppler-velocimetry study of emsemble-averaged characteristics of the turbulent near wake of a square cylinder. *J. Fluid Mech.* 304, 285–316.

E.R. Meinders, T.H. van der Meer and K. Hanjalic (1998) Local convective heat transfer from an array of wall-mounted cubes. *Int. J. Heat Mass Transfer* 41, 335–346.

H.W. Meuer and H. Gietl (2012) *Supercomputers Prestige Objects or Crucial Tools for Science and Industry?.* University of Mannheim $ Prometeus GmbH (Germany), 21 pages.

O. Nevanlinna and W. Liniger (1978) Contractive methods for stiff differential equations; Part I. *BIT* 18, 457–474.

A.K. Prasad and J.R. Koseff (1989) Reynolds number and end-wall effects on a lid-driven cavity flow. *Phys. Fluids A,* 1, 208–218.

W.C. Reynolds (1990) The potential and limitations of direct and large eddy simulations. In: J.L. Lumley (ed.) *Whither Turbulence: Turbulence at the Crossroads*. Springer Verlag, pp. 313–342.

W. Rodi, J.H. Ferziger, M. Breuer and M. Pourquié (1997) Status of Large Eddy Simulation: Results of a workshop. *ASME J. Fluids Eng.* 119, 248–262.

F.X. Trias and R.W.C.P. Verstappen (2011) On the construction of discrete filters for symmetry-preserving regularization models. *Comp. Fluids* 40, 139–148.

F.X. Trias, R.W.C.P. Verstappen, A. Gorobets, M. Soria and A. Oliva (2010) Parameter-free symmetry-preserving regularization modeling of a turbulent differentially heated cavity. *Comp. Fluids* 39, 1815–1831.

R.W.C.P. Verstappen and R.M. van de Velde (2006) Symmetry-preserving discretisation of heat transfer in a complex turbulent flow. *J. Eng. Math.* 54, 299–318.

R.W.C.P. Verstappen and A.E.P. Veldman (1996) A fourth-order finite volume method for direct numerical simulation of turbulence at higher Reynolds numbers. In: J.-A. Desideri *et al.* (eds.) *Computational Fluid Dynamics '96*. John Wiley, pp. 1073–1079.

R.W.C.P. Verstappen and A.E.P. Veldman (1997) Direct numerical simulation of turbulence at lower costs. *J. Eng. Math.* 32, 143–159.

R.W.C.P. Verstappen and A.E.P. Veldman (1998) Spectro-consistent discretization: a challenge to RANS and LES. *J. Eng. Math.* 34, 163–179.

R.W.C.P. Verstappen and A.E.P. Veldman (2003) Symmetry-preserving discretization for turbulent flow. *J. Comput. Phys.* 187, 343–368.

B.J. Vickery (1966) Fluctuating lift and drag on a long square cylinder of square cross-section in a smooth and in a turbulent stream. *J. Fluid Mech.* 25, 481–494.

P.R. Voke (1997) Flow past a square cylinder: test case LES2. In: J.P. Chollet *et al.* (eds.), *Direct and Large Eddy Simulation II*, pp. 355–373.

B. Vreman, B. Geurts and H. Kuerten (1994) Discretization error dominance over subgrid-terms in large eddy simulation of compressible shear layers in 2D. *Comm. Num. Meth. Eng.* 10, 785–790.

Appendix A

Discretization, integration and iteration

In the appendix we will give a short review of numerical fundamentals concerning space discretization, time integration and iterative solution of (sparse) systems of equations.

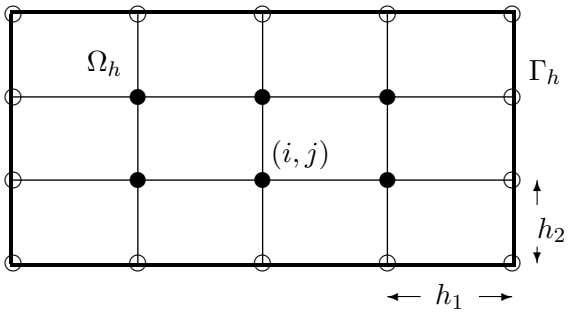
A.1 Discretization in space

Space discretization methods are often divided in four approaches:
- finite-difference methods;
- finite-volume methods;
- finite-element methods; and
- spectral methods.

This subdivision is not strict in the sense that combinations of the above approaches also appear, such as 'control-volume finite elements' or 'spectral elements'. Also, a number of discretization formulas can be derived along several of these approaches. Here, we will only make some remarks on the first two approaches. They make use of a computational grid, typically denoted by

$$\Omega_h = \{(ih_1, jh_2) | i = 0, 1, \cdots, I;\ j = 0, 1, \cdots, J\},$$

where h_1 and h_2 are the respective mesh widths.



Some notation: The discrete approximation of an unknown function ϕ is denoted by ϕ_h. The value of the grid function ϕ_h in the grid point (ih_1, jh_2) - or short (i, j) - is denoted by ϕ_{ij}. The boundary of Ω_h is denoted by Γ_h. Internal grid points are collected in Ω_h^0, the boundary points with a Dirichlet (or Robin) condition are denoted by Γ_h^D, and Γ_h^N is the set of boundary points with a Neumann condition.

A.1.1 Finite-difference methods

In a finite-difference method derivatives are approximated by combining local Taylor series such that the local truncation error becomes satisfactory in some sense, e.g. minimized between all other combinations using the same neighbouring grid points. As an example, consider the i-th grid point with its two nearest neighbours, corresponding to Taylor series

$$\phi(x_i + h) = \phi_i + h \left.\frac{\mathrm{d}\phi}{\mathrm{d}x}\right|_{x_i} + \frac{h^2}{2} \left.\frac{\mathrm{d}^2\phi}{\mathrm{d}x^2}\right|_{x_i} + O(h^3). \tag{A.1}$$

and

$$\phi(x_i - h) = \phi_i - h \left.\frac{\mathrm{d}\phi}{\mathrm{d}x}\right|_{x_i} + \frac{h^2}{2} \left.\frac{\mathrm{d}^2\phi}{\mathrm{d}x^2}\right|_{x_i} - O(h^3). \tag{A.2}$$

Subtracting (A.1) and (A.2) we obtain an estimate for the first-order derivative

$$\left.\frac{\mathrm{d}\phi}{\mathrm{d}x}\right|_{x_i} = \frac{\phi_{i+1} - \phi_{i-1}}{2h} + O(h^2). \tag{A.3}$$

which possesses a second-order accurate truncation error (cf. the term $O(h^2)$). This formula uses two points positioned symmetrically with respect to the central point. As an alternative one of these neighbours can be combined with the central point, yielding a formula that is only first-order accurate

$$\left.\frac{\mathrm{d}\phi}{\mathrm{d}x}\right|_{x_i} = \frac{\phi_i - \phi_{i-1}}{h} + O(h).$$

As we will demonstrate in Chapter 1, minimization of the local truncation error may not be the optimum criterion for selecting discretization formulas (see also the discussion in Section A.1.4).

A.1.2 Finite-volume methods

A finite-volume method starts with a physical conservation law expressing conservation of some quantity ϕ over a domain Ω with boundary Γ

$$\frac{\partial}{\partial t} \int_\Omega \phi \,\mathrm{d}\Omega = -\int_\Gamma \boldsymbol{F}(\phi) \cdot \boldsymbol{n} \,\mathrm{d}\Gamma + \int_\Omega f \,\mathrm{d}\Omega. \tag{A.4}$$

The integral over Γ describes the nett outflow of the considered quantity (given by the flux-function $\boldsymbol{F}(\phi)$), whereas f represents a source term. When the flux-function is differentiable, Gauss' divergence theorem can be applied

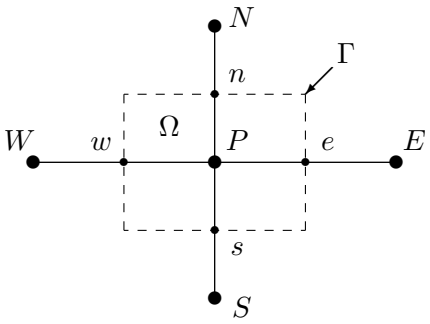
$$\int_\Omega \frac{\partial \phi}{\partial t} \,\mathrm{d}\Omega + \int_\Omega \operatorname{div} \boldsymbol{F}(\phi) \,\mathrm{d}\Omega = \int_\Omega f \,\mathrm{d}\Omega.$$

Since this holds for arbitrary small volumes Ω the conservation law can be written as a PDE in divergence form

$$\frac{\partial \phi}{\partial t} + \operatorname{div} \boldsymbol{F}(\phi) = f. \tag{A.5}$$

Note that the physical conservation law (A.4) requires less smoothness of the solution; moreover it induces the correct jump relations.

In the finite-volume method the relation (A.4) is applied to individual control volumes, surrounding the grid points. We give a typical example of a finite-volume grid in the adjacent figure. The control volumes are indicated by dashed lines, which lie halfway between the grid point $P = (x_i, y_j)$ and its neighbours N, E, S and W.



The integrals appearing in (A.4) can be approximated by e.g. the midpoint rule as

$$h_1 h_2 \frac{\partial}{\partial t}\phi_P = -h_2 \left[F_1(\phi)|_e - F_1(\phi)|_w\right] - h_1 \left[F_2(\phi)|_n - F_2(\phi)|_s\right] + h_1 h_2 f_P,$$

in which F_1 and F_2 are the components of the flux-function. After division by $h_1 h_2$ this formula can be reshaped as follows

$$\frac{\mathrm{d}}{\mathrm{d}t}\phi_h = -L_h \phi_h + f, \tag{A.6}$$

but as we will see in Section 1.5 this scaling is not always desirable. The 'curly' $\partial/\partial t$ has been replaced by a 'straight' $\mathrm{d}/\mathrm{d}t$, since in the right-hand side no spatial derivatives appear anymore. In time direction no discretization has taken place yet; we call this *semi-discretization*.

A.1.3 Properties of difference operators

Irrespective of the followed discretization approach, ultimately a spatial differential operator is approximated by a difference operator which acts on functions defined on the computational grid. Such a difference operator will be of the form

$$(L_h \phi)_{ij} = \sum_{k,l} c_{ij}^{kl}\, \phi_{kl}. \tag{A.7}$$

Definition A.1.1 The operator L_h defined in (A.7) is called *positive*[1] when

$$\sum_{k,l} c_{ij}^{kl} = 0, \qquad (i,j) \in \Omega_h^0 \cup \Gamma_h^N; \tag{A.8}$$

$$c_{ij}^{kl} \le 0, \qquad (i,j) \in \Omega_h^0 \cup \Gamma_h^N, \quad (k,l) \neq (i,j). \tag{A.9}$$

[1] Condition (A.8) implies that a positive operator can also be written in the form $(L_h\phi)_{ij} = \sum_{kl \neq ij} c_{ij}^{kl}\,(\phi_{kl} - \phi_{ij})$, with all relevant $c_{ij}^{kl} \le 0$ according to (A.9).

The above requirements need only hold in the interior Ω^0 and on Neumann boundaries Γ^N; on a Dirichtlet boundary Γ^D they are not applicable since only a central coefficient c_{ij}^{ij} is present there.

A positive operator satisfies a maximum principle, which gives smoothness to the solution: the above definition implies that the solution in a grid point is a convex combination of its neighbours:

$$\phi_{ij} = \sum_{(k,l)\neq(i,j)} -\frac{c_{ij}^{kl}}{c_{ij}^{ij}}\,\phi_{kl},$$

with all coefficients in the right-hand side non-negative and adding up to 1. *A fortiori*, when there is a direct link between adjacent grid points, the solution of $L_h\psi_h = 0$ is non-oscillatory ('no wiggles').

Theorem A.1.2 (*Maximum principle*) Let L_h, as defined in Def. A.1.1, be a positive operator and suppose $L_h\phi_h \leq 0$ in $\Omega_h^0 \cup \Gamma_h^N$. Suppose also that L_h satisfies a direct-neighbour condition

$$c_{ij}^{kl} \neq 0, \quad (k,l) \in \{(i\pm 1, j), (i, j\pm 1)\}, \quad (i,j) \in \Omega^0 \cup \Gamma^N, \tag{A.10}$$

then ϕ_h can only take its maxima on Γ_h^D (unless ϕ_h is constant).

Proof Suppose ϕ_h takes its maximum M in the grid point $(i,j) \in \Omega_h^0 \cup \Gamma_h^N$. Suppose L_h is given by (A.7). Then from

$$\sum_{k,l} c_{ij}^{kl}\,\phi_{kl} = (L_h\phi_h)_{ij} \leq 0$$

we have

$$c_{ij}^{ij}\,\phi_{ij} \leq -\sum_{(k,l)\neq(i,j)} c_{ij}^{kl}\,\phi_{kl}.$$

As the coefficients in the right-hand side are all non-positive we obtain

$$c_{ij}^{ij}\,\phi_{ij} \leq -\sum_{(k,l)\neq(i,j)} c_{ij}^{kl}\,\max_{k,l}\phi_{kl} = c_{ij}^{ij}\,\max_{k,l}\phi_{kl} = c_{ij}^{ij}\,M.$$

Because $\phi_{ij} = M$, everywhere the equality sign has to hold, from which we conclude that all neighbours ϕ_{kl} for which $c_{ij}^{kl} \neq 0$ have to be equal to M too. Because of the direct-neighbour condition (A.10), the stencils with centre in $\Omega_h^0 \cup \Gamma_h^N$ share some overlap, with which the theorem follows. □

Remark When the direct-neighbour condition (A.10) holds, whereas all other neighbour coefficients vanish, than the maximum principle also holds for local maxima.

Corollary Let L_h be a positive operator satisfying the direct-neighbour condition (A.10), and let $L_h\phi_1 = f_1$ and $L_h\phi_2 = f_2$ with $f_1 \leq f_2$. Then $\phi_1 \leq \phi_2$.

Proof Apply Th. A.1.2 to $L_h(\phi_1 - \phi_2) = f_1 - f_2$. □

Because of this property, positive operators are also called *monotone*. There exists a link with the so-called M-matrices (see Def. A.2.9), which form an important subclass of the monotone matrices (for which $A^{-1} \geq 0$).

Theorem A.1.3 When the Dirichlet part of the boundary is non-empty, then a positive operator gives rise to an M-matrix. When L_h also satisfies a direct-neighbour condition (A.10) then the matrix is irreducible.

Proof The sign requirement for an M-matrix follows immediately from the definition (A.9) of a positive operator. Further, because of (A.8) the matrix is (weakly) diagonally dominant. In Dirichlet boundary points the diagonal dominance is strict. Now a Gershgorin-type argument yields that all eigenvalues lie in the right half plane. □

Next to their monotonicity property, leading to smooth solutions as discussed above, M-matrices also possess pleasant iterative properties, e.g. Jacobi and Gauss-Seidel iteration converge (see Section A.2). Unfortunately, one has to pay a price for such favourable properties, as formulated in the following theorem.

Theorem A.1.4 (*Order barrier*) Positive (hence monotone) discretizations of a first-order derivative can maximally be first-order accurate.

Proof For a uniform one-dimensional grid with grid size h.) Suppose in grid point i we have

$$\left.\frac{\mathrm{d}\phi}{\mathrm{d}x}\right|_i \equiv \sum_k c_k \phi_{i+k}.$$

A Taylor expansion of the occurring ϕ-values results in

$$\left.\frac{\mathrm{d}\phi}{\mathrm{d}x}\right|_i \equiv \sum_k c_k \left(\phi_i + kh \left.\frac{\mathrm{d}\phi}{\mathrm{d}x}\right|_i + \tfrac{1}{2}(kh)^2 \left.\frac{\mathrm{d}^2\phi}{\mathrm{d}x^2}\right|_i + O(h^3)\right). \tag{A.11}$$

As this expression is meant to be a discrete approximation for the first-order derivative, consistency requires that the zero-th order term vanishes and that the first-order term is equal to $\mathrm{d}\phi/\mathrm{d}x$. These conditions can be satisfied. But, to achieve second-order accuracy, the second-order term in (A.11) should also vanish (note that $c_k \propto 1/h$), i.e.

$$\sum_k \tfrac{1}{2}(hk)^2 c_k = 0.$$

Alas, this is impossible when all neighbouring coefficients $c_k \leq 0$, as required for a positive operator. □

For more information on monotone discretization methods we refer to e.g. Chapter 21 of Hirsch (1990).

A.1.4 Discretization error

The above differential operators are being used for approximating (partial) differential operators. Suppose the differential equation is given by

$$L\phi = f,$$

and the difference approximation by

$$L_h \phi_h = f_h.$$

In general, the exact solution of the PDE will not satisfy the difference approximation. The difference

$$\tau_h \equiv L_h\phi - f_h = L_h(\phi - \phi_h) \tag{A.12}$$

is called the *local truncation error*. When $h \to 0$ implies that $\tau_h \to 0$, then the difference approximation is called *consistent*.

The difference $\phi - \phi_h$, between the exact solution and the discrete solution, is called the *global discretization error*. Both errors are related by

$$\phi - \phi_h = L_h^{-1}\tau_h. \tag{A.13}$$

We have *convergence* when the global discretization error vanishes upon grid refinement (i.e. when $h \to 0$). Note that the right-hand side in (A.13) consists of a product of the inverse difference operator and the local truncation error, hence smallness of the latter does not necessarily lead to an accurate discrete solution (cf. the discussion in Section 1.5).

A.2 Elementary iterative solution methods

After discretization and linearization of a partial differential equation often a linear system $Ax = b$ has to be solved. In this section we will summarize a number of iterative stationary one-step methods. These have the generic form

$$Qx^{(n+1)} = (Q - A)x^{(n)} + b, \quad n = 0, 1, \cdots, \tag{A.14}$$

where Q is a non-singular matrix. By rewriting (A.14) we obtain

$$x^{(n+1)} = Cx^{(n)} + Q^{-1}b, \quad n = 0, 1, \cdots, \qquad \text{where } C = I - Q^{-1}A. \tag{A.15}$$

The matrix C is called the *iteration matrix*.

Since the iteration error, i.e. the difference with the exact solution x, satisfies

$$x - x^{(n+1)} = C(x - x^{(n)}),$$

the iteration method (A.15) will converge if and only if $C^n \to 0$. The growth of C^n is given by

$$\|C^n\| \sim cn^{p-1}|\lambda|^{n-(p-1)}, \quad n \to \infty, \tag{A.16}$$

where λ is the modulus of the largest eigenvalue of C, and p the order of the largest Jordan block corresponding with eigenvalues of modulus λ. It easily follows that an iterative method is convergent if and only if its spectral radius $\rho(C) < 1$. The behaviour of C^n can be very irregular when defect eigenvalues are present (i.e. eigenvalues for which no eigenvector exists) which give rise to Jordan blocks.

A.2.1 Jacobi and JOR

Write the matrix A as $A = D - L - U$, in which $D = \mathrm{diag}(A)$, L a strict lower-triangular matrix and U a strict upper-triangular matrix. A simple splitting of the matrix is obtained by only leaving its diagonal in the left-hand side

$$Dx^{(n+1)} = (D - A)x^{(n)} + b,$$

which upon division by D becomes

$$x^{(n+1)} = (I - D^{-1}A)x^{(n)} + D^{-1}b. \tag{A.17}$$

The iteration matrix of this process $I - D^{-1}A = D^{-1}(L + U)$ is called the *Jacobi matrix*.

When the iteration process behaves very regularly, it may pay off to extrapolate the step $x^{(n+1)} - x^{(n)}$ as

$$x^{(n+1)} = x^{(n)} + \omega(x^* - x^{(n)}) = \omega x^* + (1 - \omega)x^{(n)},$$

where x^* is the value from (A.17). This idea is called relaxation, and ω is the *relaxation factor*. Adding relaxation to the Jacobi method (A.17) gives

$$\begin{aligned} x^{(n+1)} &= [\,\omega(I - D^{-1}A) + (1 - \omega)I\,]\,x^{(n)} + \omega D^{-1}b && \text{(A.18a)} \\ &= (I - \omega D^{-1}A)x^{(n)} + \omega D^{-1}b && \text{(A.18b)} \\ &= x^{(n)} - \omega D^{-1}(Ax^{(n)} - b). && \text{(A.18c)} \end{aligned}$$

From (A.18b) we conclude

$$\frac{1}{\omega}Dx^{(n+1)} = (\frac{1}{\omega}D - A)x^{(n)} + b,$$

which shows a splitting where D/ω is left on the diagonal in the left-hand side. This method is called JOR: Jacobi with OverRelaxation (although often $\omega < 1$, i.e. underrelaxation, gives better results). The components of $x^{(n+1)}$ can be computed independent of each other; the method is perfectly vectorizable and parallelizable.

In formulation (A.18c) we recognize the residual $Ax^{(n)} - b$. This shows that in simple words the JOR method can be expressed as: "the new guess equals the old guess plus a factor times the residual". We will use this interpretation several times in these lecture notes.

Next let us investigate the convergence behaviour of JOR. We will restrict ourselves first to the class of weakly diagonally dominant matrices, which include the M-matrices.

Theorem A.2.1 When the matrix A is irreducible and weakly diagonally dominant, i.e.

$$|a_{ii}| \geq \sum_{j \neq i} |a_{ij}| \quad \text{for all } i, \text{ with strict inequality for at least one } i,$$

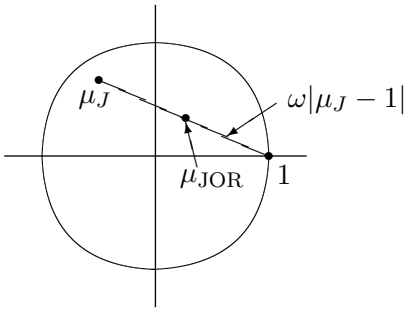
then the Jacobi method converges.

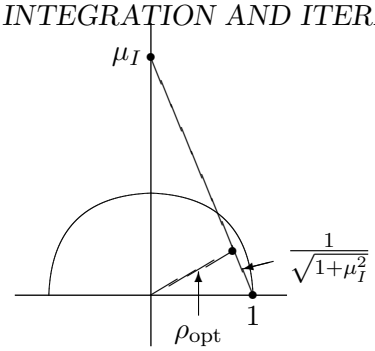
Proof Young (1971), Th. 4-2.1. □

Theorem A.2.2 When Jacobi converges, then also JOR with $0 < \omega \leq 1$ converges.

Proof From (A.18a), the iteration matrix of JOR is given by $\omega(I - D^{-1}A) + (1 - \omega)I$. Its eigenvalues μ_{JOR} are related the eigenvalues μ_J of the Jacobi matrix by

$$\mu_{\text{JOR}} = \omega\,\mu_J + 1 - \omega = 1 + \omega\,(\mu_J - 1).$$





Observe that this corresponds with inter(extra)polation between the points 1 and μ_J in the complex plane. The figure reveals that as soon as Jacobi converges, i.e. all Jacobi eigenvalues lie within the unit circle, all eigenvalues for JOR with $0 < \omega \leq 1$ also lie within the unit circle. This establishes the proof. □

The above proof reveals that the eigenvalues of the Jacobi matrix $I - D^{-1}A$ have to lie on the left side of the line $\Re\mathrm{e}\,\mu = 1$. Thus, for complex eigenvalues we have in general

Theorem A.2.3 The JOR method converges for sufficiently small $\omega(>0)$ if and only if the matrix $D^{-1}A$ is *positive*-stable (i.e. all eigenvalues of $D^{-1}A$ are located in the right halfplane).

In many cases the eigenvalues of the Jacobi matrix lie outside the unit circle, hence the Jacobi method will no longer converge. Often the eigenvalues lie far up on the imaginary axis. Let the largest eigenvalue be given by $\mu_J = i\mu_I$, with μ_I real. In the above figure we observe that the JOR method converges for $0 < \omega < \omega_{\rm JOR,max}$ where $\omega_{\rm JOR,max} = \frac{2}{1+\mu_I^2}$. The optimum relaxation factor is given by

$$\omega_{\rm JOR,opt} = \frac{1}{1+\mu_I^2}, \text{ with spectral radius } \rho_{\rm JOR,opt} = \frac{\mu_I}{\sqrt{1+\mu_I^2}}. \tag{A.19}$$

A.2.2 Gauss-Seidel and SOR

In the Gauss-Seidel method also the lower-triangular part of the matrix is kept in the left-hand side, hence

$$(D - L)x^{(n+1)} = Ux^{(n)} + b.$$

Its relaxation version SOR (Successive OverRelaxation) reads

$$(\frac{1}{\omega}D - L)x^{(n+1)} = \left[(\frac{1}{\omega} - 1)D + U\right] x^{(n)} + b,$$

with iteration matrix

$$C_\omega = (D - \omega L)^{-1}[(1-\omega)D + \omega U]. \tag{A.20}$$

For consistently ordered matrices (see Young, 1971) a relation exists between the eigenvalues λ of C_ω and the eigenvalues μ_J of the Jacobi matrix. This relation reads

$$(\lambda + \omega - 1)^2 = \omega^2\mu_J^2\lambda.$$

A special case is $\omega = 1$, when this relation reduces to $\lambda = \mu_J^2$. This implies that for consistently ordered matrices Gauss-Seidel converges if and only if Jacobi converges. More precise,

for such matrices Gauss-Seidel converges/diverges twice as fast as Jacobi. As a corollary, Gauss-Seidel converges for consistently ordered M-matrices.

For a large class of matrices SOR can be made convergent, as shown in the next theorem.

Theorem A.2.4 Let A be a consistently ordered matrix. Then SOR converges for sufficiently small $\omega(>0)$ if and only if $D^{-1}A$ is positive-stable.

Proof Young (1971), Th. 6-4.2. □

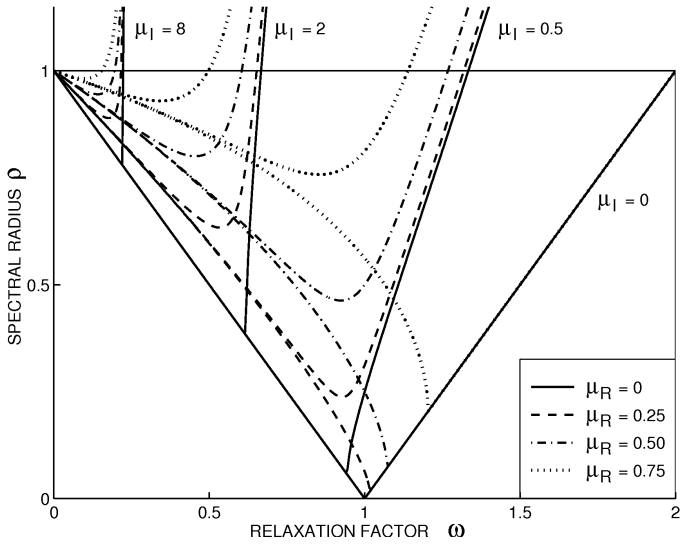


Figure A.1: *Relation between spectral radius, relaxation factor and Jacobi eigenvalue* $\mu = \mu_R + i\mu_I$.

In Figure A.1 we show the spectral radius of the SOR matrix (A.20) for various choices of the determining Jacobi eigenvalues. We note that for a symmetric matrix A (for which the Jacobi spectrum is real), SOR converges if and only if A is positive definite and $0 < \omega < 2$. The optimum relaxation factor is given by (see Young (1971) Th. 4-3.3 and 4-3.6)

$$\omega_{\mathrm{SOR,opt}} = \frac{2}{1 + (1 - \mu_R^2)^{1/2}}, \quad \text{with } \rho_{\mathrm{SOR,opt}} = \omega_{\mathrm{SOR,opt}} - 1.$$

When the Jacobi eigenvalues are complex, than the ω-interval for which convergence occurs becomes smaller. For purely imaginary Jacobi spectra, with extremal eigenvalues $\mu_J = i\mu_I$, it can be shown (Botta and Veldman 1982) that SOR converges if and only if

$$0 < \omega < \omega_{\mathrm{SOR,max}} \text{ with } \omega_{\mathrm{SOR,max}} = \frac{2}{1 + \mu_I}.$$

Hence, when $\mu_I > 1$ Gauss-Seidel no longer converges (in fact, Jacobi does neither). The optimum relaxation factor is given by

$$\omega_{\mathrm{SOR,opt}} = \frac{2}{1 + (1 + \mu_I^2)^{1/2}}, \text{ with } \rho_{\mathrm{SOR,opt}} = 1 - \omega_{\mathrm{SOR,opt}}. \tag{A.21}$$

It follows that $\omega_{\mathrm{SOR,opt}} < 1$, i.e. *under*relaxation is desired (and for $\mu_I > 1$ even necessary).

A.2.3 Some definitions and theorems on eigenvalues

In this section we present some definitions and theorems on the location of matrix eigenvalues.

Lemma A.2.5 (*Bendixson*) Let $A = R + i\,S$ be a general (complex) matrix, where $2R = A + A^*$ is its symmetric part, and $2iS = A - A^*$ its skew-symmetric part (hence R and S are Hermitian[2]). Let ρ_m and ρ_M be the smallest and largest eigenvalues of R, respectively; similarly σ_m and σ_M w.r.t. S. Then the eigenvalues of A are located in a rectangle defined by the lower-left and upper-right corners (ρ_m, σ_m) and (ρ_M, σ_M).

Proof (from Householder 1964, p. 79) Let x be a normed eigenvector (i.e. $x^*x = 1$) corresponding to an eigenvalue $\lambda = a + i\,b$ of A. Then

$$a + i\,b = \lambda = \lambda x^*x = x^*Ax = x^*Rx + i\,x^*Sx,$$

hence $a = x^*Rx$ and $b = x^*Sx$. Since both R and S are Hermitian matrices their eigenvalues are real and one has

$$\rho_m \leq x^*Rx \leq \rho_M \quad \text{for any } x;$$

a similar relation holds for S. Now the theorem follows. □

Definition A.2.6 *(e.g. Horn and Johnson 1991, p. 92)* A matrix A is called *positive stable* if all of its eigenvalues are located in the right (i.e. positive) halfplane $\Re\mathrm{e}\ \lambda(A) > 0$.

Definition A.2.7 *(e.g. Young 1971, p. 24)* A matrix A is called *positive real* if its symmetric part $A + A^*$ is positive definite[3].

The property 'positive stable' is somewhat weaker than the property 'positive real', as follows from the next lemma.

Lemma A.2.8 Let A be a positive-real matrix. Then for each positive-definite matrix Q the product QA is positive stable; in particular, the matrix A itself is positive stable.

Proof We first prove that $P \equiv Q^{1/2}AQ^{1/2}$ is positive real. This follows from

$$\begin{aligned}((x, (P + P^T)x)) &= ((x, (Q^{1/2}AQ^{1/2} + Q^{1/2}A^TQ^{1/2})x)) \\ &= ((x, Q^{1/2}(A + A^T)Q^{1/2}x)) = ((Q^{1/2}x, (A + A^T)Q^{1/2}x)) > 0\end{aligned}$$

for all $x \neq 0$. Subsequently we prove that P is positive stable. Hereto let λ be an eigenvalue of P with corresponding eigenvector y. Then we have

$$\begin{aligned}2\,\Re\mathrm{e}\,\lambda\,\|y\|^2 &= (\lambda + \overline{\lambda})\|y\|^2 = ((\lambda y, y)) + ((y, \lambda y)) \\ &= ((Py, y)) + ((y, Py)) = ((y, (P + P^T)y)) > 0,\end{aligned}$$

from which we conclude $\Re\mathrm{e}\ \lambda > 0$. The theorem now follows since $QA = Q^{1/2}PQ^{-1/2}$ possesses the same eigenvalues as P. □

[2] A complex matrix M is called Hermitian (or self-adjoint) if it is equal to its conjugate transpose M^*. Hermitian matrices can be understood as the complex extension of real symmetric matrices.

[3] Alternative definition (Wachspress 1966, p. 9): A matrix A is called positive real if $(x, Ax) > 0$ for all real $x \neq 0$. For comparison, when $(x, Ax) > 0$ for all complex $x \neq 0$ the matrix is positive definite.

Remark That a positive-real matrix is positive stable also follows directly from Lemma A.2.5.

An important example of positive-stable matrices is formed by the so-called M-matrices.

Definition A.2.9 A matrix $A \equiv (a_{ij})$ is called an *M-matrix* when
1. its diagonal elements are positive $a_{ii} > 0$, and its non-diagonal elements are non-positive $a_{ij} \leq 0$, $i \neq j$;
2. the matrix is positive stable, i.e. all eigenvalues of A lie in the right halfplane $\Re\mathrm{e}\, \lambda(A) > 0$.

Many properties of M-matrices have been collected in Chapter 2.5 of Horn and Johnson (1991). An important property is that the inverse of an M-matrix A satisfies $A^{-1} \geq 0$, hence it is monotone; see also the RUG lecture notes "Computational Methods of Science".

Another useful lemma to compute the eigenvalues of a tri-diagonal matrix is

Lemma A.2.10 The eigenvalues of a tri-diagonal $n \times n$ matrix with entries $\{a, b, c\}$ are given by

$$\lambda = b + 2\sqrt{ac}\,\cos\left(\frac{i\,\pi}{n+1}\right), \quad i = 1, \cdots, n.$$

In fact, the above lemma describes one of the rare occasions where eigenvalues can be computed analytically.

A.3 Integration in time

Let us now proceed from (A.6) where the spatial derivatives have been discretized with one of the methods from the preceding section

$$\frac{\mathrm{d}\phi}{\mathrm{d}t} + L_h\phi = 0. \tag{A.22}$$

To integrate this equation in time, discrete time levels $t_n = n\,\delta t$ are introduced. Hereafter a recipe is used to compute the newest time level. As an example, explicit time integration (also known as the forward Euler method) is given by

$$\frac{\phi^{(n+1)} - \phi^{(n)}}{\delta t} = -L_h\phi^{(n)}, \tag{A.23}$$

where the spatial-derivative term L_h is evaluated at the 'old' time level. In the Crank–Nicolson method this term is evenly divided between the old and the new time level

$$\frac{\phi^{(n+1)} - \phi^{(n)}}{\delta t} = -\tfrac{1}{2}\left(L_h\phi^{(n+1)} + L_h\phi^{(n)}\right), \tag{A.24}$$

whereas the backward Euler method evaluates it at the new time level only.

Many more methods are feasible; we refer to textbooks on solving differential equations (e.g. Lambert 1973) for an overview of the large amount of time-integration methods that have been designed. Anyway, a generic time step (using only one previous time level) will look like

$$\phi^{(n+1)} = A(\delta t)\,\phi^{(n)}. \tag{A.25}$$

Such a time step will have a certain formal asymptotic accuracy, e.g. forward and backward Euler are first-order accurate in time, whereas Crank–Nicolson is second-order accurate.

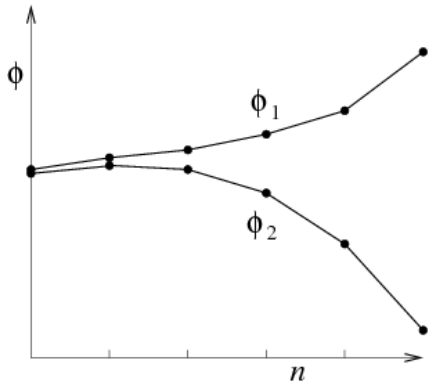
A.3.1 Stability

Next to accuracy, the stability of the process (A.25) when n grows is of concern. Stability may put restrictions on the time step that are more stringent than those put by accuracy (especially for stiff equations there can be a large discrepancy). The notion of stability is closely related to the well-posedness of the discrete problem, i.e. a solution should depend continuously on the parameters defining the problem.

As an example, well-posedness with respect to initial conditions requires that two solutions $\phi_1(t)$ and $\phi_2(t)$, starting 'close' to each other at $t = 0$ remain 'close' to each other: the situation in the adjacent figure is not what one would like to see. In mathematical terms, for any time interval $0 \leq t \leq T$ there should exist a constant K_T such that

$$\|\phi_1(t) - \phi_2(t)\| \leq K_T \, \|\phi_1(0) - \phi_2(0)\| \quad \text{for all} \quad t \leq T.$$

When the underlying equation is linear, the above relation also holds for the solution itself (not just for the difference between two solutions).



Stability is often a confusing notion, therefore we will pay some attention to it here. At first we have to further specify the type of process we want to investigate. There are two ways in which n can grow, corresponding with two types of stability:

- absolute stability, also known as stepwise stability,
- zero-stability or pointwise stability.

Absolute stability is concerned with keeping the time step δt fixed while $n \to \infty$ (i.e. $T = \infty$ in the above well-posedness definition). This notion of stability is relevant when the solution of the steady problem in the limit $t \to \infty$ is pursued. The time-integration method has become an iteration process for finding the solution of $L_h\phi_h = 0$, where the time step plays a role as relaxation parameter. Absolute stability demands that the subsequent time levels remain bounded, i.e.

$$\|A^n(\delta t)\| \leq K \quad \text{for all } n. \tag{A.26}$$

For the process to converge when time proceeds a stronger condition has to hold, viz. $\|A^n(\delta t)\| \to 0$ for $n \to \infty$. We will see later (cf. Section 1.8) that absolute stability cannot guarantee that during the process (i.e. for finite t) no 'strange things' can happen.

Zero-stability is concerned with finding the solution on a fixed and finite time interval $[0, T]$. In this case we let the time step δt go to zero (hence its name), which makes it natural also to let the mesh size h go to zero. During this process, when time step and mesh size simultaneously approach zero, the discrete solution should converge to the continuous one. The number of time steps grows since the time steps decreases, but the total number of time steps remains bounded for non-zero time step, namely it is bounded by $T/\delta t$. During this limit process the discrete formulation should remain well-posed uniformly in δt (with

$0 < \delta t \leq \delta t_{max}$). I.e. there has to exist a constant K_T, independent of n, δt and h, such that

$$\|A^n(\delta t)\| \leq K_T \quad \text{for} \quad 0 < n \leq T/\delta t. \tag{A.27}$$

The Equivalence Theorem of Lax applies to this situation: for a consistent discretization, zero-stability is equivalent with convergence. For a proof see Richtmyer and Morton (1967).

Thus zero-stability guarantees that for sufficiently small δt the discrete solution becomes a good approximation of the continuous solution. However it gives no clue about how small the time step has to be chosen to attain a certain accuracy. A guide line for selecting an accurate δt can be obtained through the notion of practical stability introduced by Richtmyer and Morton. This notion is sort of an intermediate between well-posedness and accuracy.

Practical stability demands that the growth of the discrete solution, characterized by $\|A(\delta t)\|$, is similar to the growth of the continuous solution. Since $\|A(\delta t)\|$ often is difficult to determine, Richtmyer and Morton (1967) formulate their guide line in terms of Fourier analysis:

> no discrete Fourier component may grow (significantly) faster than the maximum growth of the exact solution.

We have to realize that Fourier analysis does not take into account the influence of boundary conditions, non-equidistant grids, non-constant coefficients and non-linearity of the equations. Nevertheless, as we will see in Section 1.8, for practical situations a useful guide line is provided for selecting δt and h. For equations of which the solution does not grow, practical stability leads to the (absolute) requirement $|g(\theta)| \leq 1$, $0 \leq \theta \leq 2\pi$, where $g(\theta)$ is the amplification factor of the Fourier component with frequency θ (see below). Historic remark: Fourier analysis of difference equations has been introduced by John von Neumann; especially he considered the zero-stability-like variant (see page 107).

Remark On the one hand (because of the condition 'for all n') absolute stability is a stronger requirement than zero-stability. On the other hand (because K is allowed to depend on h and δt) it is a weaker requirement. We will see in Section 1.8 that neither notion follows from the other.

A.3.2 Matrix analysis

The requirements put by the various types of stability can be expressed in terms of the spectrum of the operator $A(\delta t)$ describing the time evolution. In Section A.2 we mention that

$$\|A^n\| \sim cn^{p-1}[\rho(A)]^{n-(p-1)}, \qquad n \to \infty, \tag{A.28}$$

where $\rho(A)$ is the spectral radius, and p the order of the largest Jordan block belonging to the eigenvalues with $|\lambda| = \rho(A)$. When A is non-defect, i.e. A is diagonalizable, then $p = 1$. This induces

Theorem A.3.1 The process (A.25) is absolutely stable if and only if

$$\rho(A) < 1 \ \vee \ [\rho(A) = 1 \wedge |\lambda| = 1 \text{ non defect }]. \tag{A.29}$$

When $\rho(A) < 1$ then $\|A^n\| \to 0$ for $n \to \infty$, but during the transient $\|A^n\|$ can become large (see Section 1.8).

The investigation of zero-stability is more complicated as the boundedness of $\|A^n\|$ does not have to hold for all n (n always remains finite and the asymptotic behaviour (A.28) does not contain sufficient information). To keep control over the situation we will restrict ourselves to non-defect matrices.

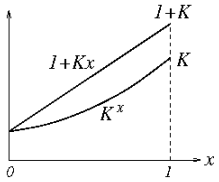
Theorem A.3.2 When A is non-defect, the following three properties are equivalent:

1. the process (A.25) is zero-stable, i.e. (A.27) holds;
2. the spectral radius of A satisfies the Von Neumann condition
$$\rho \leq 1 + O(\delta t), \tag{A.30}$$
where the order constant is independent of δt and h;
3. at each time $t > 0$ we have
$$\|\phi^{(n)}\| \leq k e^{ct} \|\phi^{(0)}\|, \tag{A.31}$$
where k and c are independent of δt and h.

Proof $1 \Rightarrow 2$: Suppose (A.27) holds. Then $\rho^n(A) = \rho(A^n) \leq \|A^n\| \leq K_T$, $0 < n \leq T/\delta t$. Now we can employ that fact that this only has to hold for a limited number of time steps. The most 'challenging' situation is for the largest value of n, i.e. $n = T/\delta t$, leading to the requirement $\rho^{T/\delta t} \leq K_T$. This can be rewritten as

$$\rho \leq K_T^{\delta t/T} \leq 1 + K_T\, \delta t/T = 1 + O(\delta t),$$

since for $0 \leq x \leq 1$ we have $K^x \leq 1 + Kx$ (see figure).



$2 \Rightarrow 3$: Next let $\rho \leq 1 + c\delta t$, and assume $c \geq 0$. Then from $1 + x \leq e^x$ we have for $n = t/\delta t$

$$\rho^n \leq (1 + c\delta t)^n \leq e^{c\delta t n} = e^{ct}.$$

Because of $\|\phi^{(n)}\| \leq \|A^n\| \|\phi^{(0)}\|$ and the non-defect variant of (A.28) the estimate (A.31) follows.

$3 \Rightarrow 1$: By choosing K_T equal to $\max(1, ke^{cT})$ we obtain (A.27). □

Remark The estimate (A.31) shows that zero-stability is the discrete equivalent of the notion well-posedness for continuous problems (cf. RUG lecture notes Partial Differential Equations, or any other mature book on PDE).

A.3.3 Positive time-integration schemes

Similar to the definition of positive (monotone) operators for steady problems (cf. Def. A.1.1), for unsteady problems the notion of positivity can be introduced. In this case the 'central' unknown is $\phi_i^{(n+1)}$; its coefficient has to be positive, whereas all other coefficients should be non-positive. In a somewhat different notation than used in Def. A.1.1, the following definition emerges for linear time-integration methods.

Definition A.3.3 Under the consistency condition $\sum_k c_i^k = 1$, the (explicit) time-integration scheme

$$\phi_i^{(n+1)} = \sum_k c_i^k \phi_k^{(n)} \equiv A\phi^{(n)}$$

is called positive when all coefficients $c_i^k \geq 0$.

Harten, Hyman and Lax (1976) give a generalization of this definition to non-linear schemes. Further, for implicit schemes the definition is similar.

Because the values at the new time level are a convex combination of those at the old time level, an immediate consequence of positivity is that global extrema cannot grow

$$\min_k \phi_k^{(n)} \leq \phi_i^{(n+1)} \leq \max_k \phi_k^{(n)}.$$

In more abstract terms, this is equivalent to saying that a positive time-integration scheme is (absolutely and zero-)stable in the maximum norm ($\|A\|_\infty = \max_i \sum_k |c_i^k| = 1$).

When the steady operator is positive it is not difficult to construct a positive time-integration scheme.

Theorem A.3.4 When the steady operator L_h defined in (A.7) is positive, then forward Euler time integration of (A.22) gives rise to a positive scheme if and only if the time step satisfies $\delta t \leq 1/c_{ij}^{ij}$.

Proof The consistency condition is an immediate consequence of (A.8). Further, by rearranging terms we have

$$\phi_{ij}^{(n+1)} = (1 - c_{ij}^{ij}\delta t)\phi_{ij}^{(n)} - \delta t \sum_{(k,l)\neq(i,j)} c_{ij}^{kl}\phi_{kl}^{(n)}.$$

Because of (A.9), only the coefficient of $\phi_{ij}^{(n)}$ requires some attention, leading to the indicated requirement on the time step. □

A.3.4 Fourier analysis

As it is difficult to determine the norm or the eigenvalues of A, a stability analysis is often carried out in terms of Fourier analysis. Here the solution ϕ_h is decomposed into Fourier components $e^{i\omega x}$. The interval $[0, L]$ is discretized with mesh width $h = L/K$ and grid points $x_k = kh = kL/K \quad (k = 0, \ldots, K)$. On this interval, the Fourier components with $\omega = 2\pi\alpha/L \quad (\alpha = 0, 1, \ldots)$ are orthogonal. Rewriting the exponent via

$$\omega x = \omega kh = \frac{2\pi\alpha}{L}\,\frac{kL}{K} = \frac{2\pi\alpha}{K}\,k \equiv \theta k,$$

some analysis reveals that on a uniform grid the Fourier modes

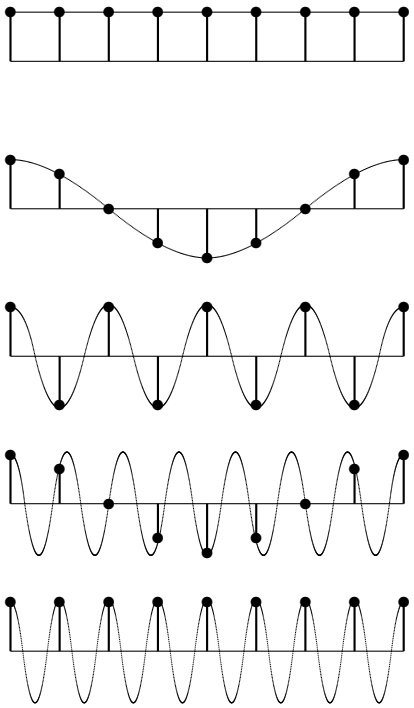
$$e^{i\theta_\alpha k} \text{ with } \theta_\alpha = \frac{2\pi\alpha}{K} \quad , \quad \alpha = 0, 1, \ldots, K-1. \tag{A.32}$$

are orthogonal, and sufficient in number to describe any periodic grid function defined in the 'only' K independent discrete points, i.e. they form an orthogonal basis.

The figure below shows the real part of some discrete basis functions, where one has to recall that information between the grid points is not visible. In this example with $K = 8$ the function with $\alpha = 4$ (i.e. $\theta_\alpha = \pi$) corresponds with the highest frequency. Larger values of α appear to possess in the grid points a lower frequency. In particular we have

$$f_{\alpha+K}(k) = f_\alpha(k).$$

This phenomenon where on the discrete grid higher frequencies cannot be distinguished from lower frequencies is known as *aliasing*. Already in the pioneering film age this phenomenon was known: carriage wheels that seem to be standing still or turn backward. The finite number of frames is responsible for this.



$\alpha = 0$;

$\alpha = 1$;

$\alpha = 4 = K/2$, $\theta_\alpha = \pi$*: as a grid function this one has the highest frequency;*

$\alpha = 7 = K - 1$*: the real part equals that of* $\alpha = 1$*, the imaginary part is different however;*

$\alpha = 8 = K$, $\theta_\alpha = 2\pi$*: the same grid function as the one at* $\alpha = 0$*, and hence not to be included as a basis function.*

Amplification factor

Substitution of the Fourier decomposition

$$\phi_h^{(n)}(x_k) = \sum_{\alpha=0}^{K-1} c_\alpha^{(n)}\, e^{ik\theta_\alpha},$$

into the discretized differential equation will ultimately result in a relation expressing the amplitude at the new time level $c^{(n+1)}$ in terms of the previous amplitude $c^{(n)}$. Their quotient

$$g(\theta_\alpha) \equiv c_\alpha^{(n+1)}/c_\alpha^{(n)} \tag{A.33}$$

is called the *amplification factor*.

As before, both step wise (absolute) and point wise (zero) stability variants can be distinguished. It can be shown that (Fourier) absolute stability corresponds with

$$\max_{\alpha} |g(\theta_\alpha)| \leq 1, \tag{A.34}$$

whereas the integration method is (Fourier) zero-stable if and only if the Von Neumann condition

$$\max_{\theta\in[0,2\pi]} |g(\theta)| \leq 1 + O(\delta t) \tag{A.35}$$

holds, where the order constant should be independent of h.

It is remarked, that the above analysis is exact when four properties are satisfied: *i)* the equation is linear; *ii)* the equation has constant coefficients; *iii)* the boundary conditions are periodic; *iv)* the grid is uniform. In other situations, the analysis is only approximate.

For practical stability a stronger requirement is set for the order constant, namely it should be related to the analytic growth of the Fourier components in the continuous problem. For diffusive problems, not allowing any growth, this leads to the requirement

$$\max_{\theta\in[0,2\pi]} |g(\theta)| \leq 1. \tag{A.36}$$

Compared with (A.34) we must stress that we are dealing with a different situation: $\delta t \to 0$ versus δt fixed. The optical similarity between (A.34) and (A.36) gives rise to much confusion about stability in the existing literature.

The Fourier symbol

The amplification factor is built from the amplification of the individual derivatives in the PDE. For example, a first-order derivative $\mathrm{d}\phi/\mathrm{d}x$ discretized as

$$\frac{\phi_{k+1} - \phi_{k-1}}{2h}$$

when applied to $\phi = e^{i\theta k}$ gives a contribution (after division by $e^{i\theta k}$)

$$\frac{e^{i\theta} - e^{-i\theta}}{2h} = \frac{i}{h}\,\sin\theta;$$

a second-order derivative $\mathrm{d}^2\phi/\mathrm{d}x^2$ discretized centrally gives a contribution

$$\frac{e^{i\theta} - 2 + e^{-i\theta}}{h^2} = \frac{2}{h^2}\,(\cos\theta - 1).$$

The expressions in the right-hand sides are called the *discrete Fourier symbol* of the difference quotients. They should form approximations of the corresponding Fourier symbols $P(i\omega)$ of the differential operators $P(\partial/\partial x)$. These symbols describe the evolution of a discrete Fourier component $e^{i\theta k}$ and a continuous Fourier component $e^{i\omega x}$, respectively

$$P_h(\partial/\partial x)e^{i\theta k} = P_f e^{i\theta k} \quad \text{versus} \quad P(\partial/\partial x)e^{i\omega x} = P(i\omega)e^{i\omega x}.$$

Referring to one of the above examples, the operator $\partial/\partial x$ possesses the Fourier symbol $i\omega$. Its centrally discretized counterpart possesses the discrete Fourier symbol

$$\frac{i}{h}\sin\theta = \frac{i}{h}\sin\omega h \sim i\omega + O(\omega^3 h^2).$$

When $\omega h = \theta$ is small, the discrete Fourier symbol approaches its continuous counterpart. The same holds for the discrete Fourier symbol of the second-order derivative

$$\frac{2}{h^2}(\cos\theta - 1) = \frac{2}{h^2}(\cos\omega h - 1) \sim -\omega^2 + O(\omega^4 h^2).$$

The criterion 'ωh is small' implies that the mesh width h has to be small with respect to to the wave length $2\pi/\omega$ of the Fourier component. For these relatively long waves the continuous and discrete evolution are more or less the same. For short waves, with ($\theta \approx \pi$), the discrete evolution differs considerably from the continuous evolution, thus jeopardizing accuracy.

Below we will give the relation between Fourier analysis for the continuous problem and the corresponding (semi-)discrete versions. In all three cases the solution is decomposed in Fourier components $e^{i\omega x}$ and $e^{i\theta k}$, respectively, where $\omega h = \theta$.

continuous	**semi-discrete**	**discrete** - forward Euler
$\partial\phi/\partial t = P(\partial/\partial x)\phi$	$d\phi_h/dt = P_h(\partial/\partial x)\phi_h$	$\frac{\phi_h^{(n+1)}-\phi_h^{(n)}}{\delta t} = P_h(\partial/\partial x)\phi_h^{(n)}$
$\phi(t) = a(t)e^{i\omega x}$	$\phi_h(t) = a_h(t)e^{i\omega x}$	$\phi_h^{(n)}(x_k) = c_\theta^{(n)} e^{i\theta k}$
$da/dt = P(i\omega)a$	$da_h/dt = P_f a$	$\frac{c_\theta^{(n+1)}-c_\theta^{(n)}}{\delta t} = c_\theta^{(n)} P_f$
$a(t) = a(0)e^{P(i\omega)t}$	$a_h(t) = a_h(0)e^{P_f t}$	$g(\theta) \equiv c_\theta^{(n+1)}/c_\theta^{(n)} = 1 + \delta t P_f$
$a(t+\delta t)/a(t) = e^{P(i\omega)\delta t}$	$a(t+\delta t)/a(t) = e^{P_f \delta t}$	
$\approx 1 + P(i\omega)\delta t$	$\approx 1 + P_f \delta t$	

Another time integration method leads to another amplification factor. For example backward Euler leads to $g(\theta) = (1 - \delta t P_f)^{-1}$, whereas for Crank-Nicolson $g(\theta) = (1 + \frac{1}{2}\delta t P_f)/(1 - \frac{1}{2}\delta t P_f)$. These expressions are approximately equal for small δt, but quite different for large δt.

For the semi-discrete version to be a good approximation of the continuous problem, the discrete Fourier symbol P_f has to be a good approximation of the continuous Fourier symbol $P(i\omega)$. Subsequently, for the discrete time integration to be a good approximation of the exact time integration, the amplification factor $g(\theta)$ in turn has to be a good approximation of the analytical growth $e^{P_f\ \delta t}$. Combined this leads to a desire to have

$$g(\theta) \approx e^{P_f \delta t} \approx e^{P(i\omega)\delta t}. \tag{A.37}$$

By comparing modulus and argument, respectively, of the left-hand side and right-hand side, statements can be made on the diffusive and dispersive character of the discrete approximation. Hereto, we repeat the analytical solution for a Fourier component

$$e^{P(i\omega)t + i\omega x} = e^{\mathrm{Re}\,P(i\omega)t} e^{i(\mathrm{Im}\,P(i\omega)t + \omega x)}.$$

It is observed that analytically diffusion is governed by $\mathrm{Re}\, P(i\omega)$, whereas the propagation speed (phase velocity) is given by $-\omega^{-1}\mathrm{Im}\, P(i\omega)$. Hence, from (A.37) one may compare

$$|g(\theta)| \longleftrightarrow \mathrm{e}^{\mathrm{Re}\, P(i\omega)\delta t} \quad \text{and} \quad c_h \equiv -\frac{1}{\omega\,\delta t}\arg g(\theta) \longleftrightarrow -\frac{1}{\omega}\,\mathrm{Im}\, P(i\omega). \tag{A.38}$$

Example

The simple-wave equation

$$\phi_t + c\,\phi_x = 0 \tag{A.39}$$

discretized using forward Euler in time and upwind in space gives

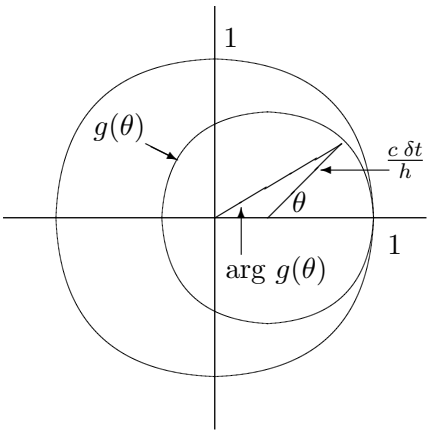
$$\frac{\phi_k^{(n+1)} - \phi_k^{(n)}}{\delta t} + c\,\frac{\phi_k^{(n)} - \phi_k^{(n)}}{h} = 0. \tag{A.40}$$

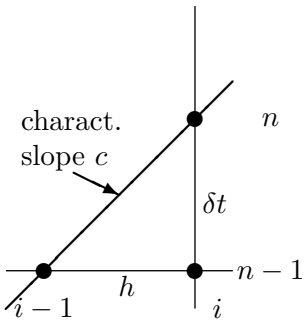
Its discrete Fourier symbol reads $-c(1-e^{-i\theta})/h$ and the amplification factor becomes

$$g(\theta) = 1 - \frac{c\,\delta t}{h}(1 - e^{-i\theta}). \tag{A.41}$$

The locus of the amplification factor is a circle with centre $1 - \frac{c\,\delta t}{h}$ and radius $\frac{c\,\delta t}{h}$. Hence (A.40) is Fourier-stable if and only if

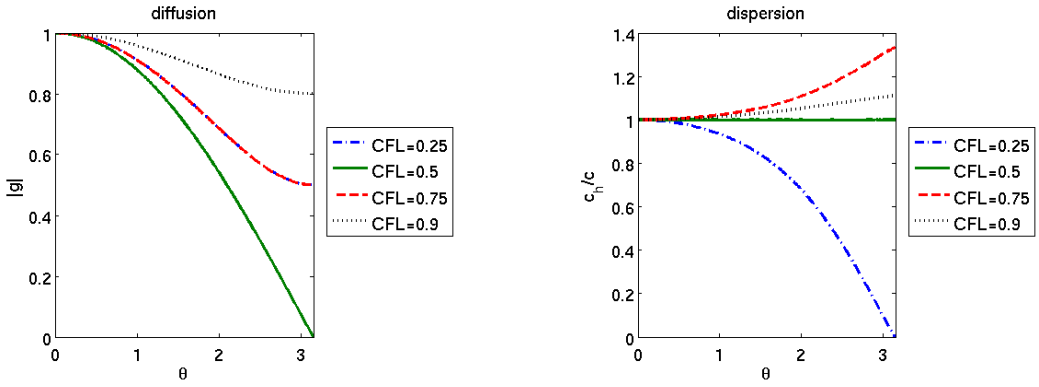
$$CFL \equiv \frac{c\,\delta t}{h} \leq 1.$$





The expression in the left-hand side is called the CFL-number, after Courant, Friedrichs and Lewy, who in 1928 for the first time brought up the notion of (numerical) stability. This requirement means that a numerical method can never be smarter than the physics, in the sense that one can never take larger time steps than allowed by the region of determinacy.

In this example the analytic solution is given by waves of the form $\mathrm{e}^{i\omega(x-ct)}$. These waves have no diffusion, hence the analytical Fourier symbol is purely imaginary, and the right-hand side of (A.37) has modulus 1. The left-hand side $g(\theta)$ possesses a modulus less or equal to 1 (at least in the stable range), with a minimum for $\theta = \pi$, i.e. the discrete approximation possesses diffusion: the shortest wave on the grid is damped most. Only when $CFL = 1$ no



damping occurs; see the left-hand figure below, where $|g|$ is plotted as a function of θ.

Further, the waves have a propagation speed (phase velocity) equal to c. From (A.38), their discrete propagation speed is given by

$$c_h = -(\omega\ \delta t)^{-1}\arg\ g(\theta),$$

with $g(\theta)$ given by (A.41). The right-hand figure shows the ratio c_h/c as a function of θ. It is observed that the discrete propagation speed differs from the analytic propagation speed, dependent on the wave number $\omega = \theta/h$. Herewith the discrete approximation introduces dispersion. When $CFL < 0.5$ the discrete waves move too slow, when $CFL > 0.5$ they move too fast. Only in the cases $CFL = 0.5$ and $CFL = 1$ we have $c_h = c$.

We saw already that for $CFL = 1$ the scheme is also free of dissipation. Note that for this special value of the CFL-number the discretized formula (A.40) reduces to

$$\phi_i^{(n+1)} = \phi_{i-1}^{(n)}.$$

This is exact since in this case the characteristic passes through both points involved (see the right-hand figure on top of page 109).

A.3.5 The modified equation

Finally we present in this section an alternative way to determine the dissipative and dispersive properties of a discrete equation. Hereto we take the reverse way. Starting form a discrete equation we trace back which differential equation does correspond; we call this the *modified equation*. This equation can be interpreted through physical insight or analytic-mathematical techniques.

As an example we again treat the simple-wave equation (A.39), discretized with forward Euler in time and upwind in space (A.40). We express all appearing grid values through Taylor series around $\phi_i^{(n)}$, hence

$$\begin{aligned} \phi_i^{(n+1)} &= \phi_i^{(n)} + \delta t\,\phi_t + \tfrac{1}{2}(\delta t)^2\,\phi_{tt} + \cdots, \\ \phi_{i-1}^{(n)} &= \phi_i^{(n)} - h\,\phi_x + \tfrac{1}{2}h^2\,\phi_{xx} + \cdots. \end{aligned}$$

Substitution in (A.40) yields

$$\phi_t + c\,\phi_x = \tfrac{1}{2}(c\,h\,\phi_{xx} - \delta t\,\phi_{tt}) + \cdots . \tag{A.42}$$

All time derivatives are transformed into spatial derivatives by using the differential equation (A.39), such as in

$$\phi_{tt} = (\phi_t)_t = (-c\phi_x)_t = -c(\phi_t)_x = -c(-c\phi_x)_x = c^2\phi_{xx}.$$

Now (A.42) transforms to the modified equation

$$\phi_t + c\,\phi_x = \tfrac{1}{2}c\,(1 - CFL)\,h\,\phi_{xx} + \cdots . \tag{A.43}$$

The right-hand side contains the difference with the equation that we wanted to solve. We recognize a diffusive term. From PDE theory it is known that this equation is stable/unstable for positive/negative diffusion, which (for $c > 0$) physically suggests the stability requirement $0 \leq 1 - CFL$. And this is indeed the correct numerical criterion.

A.4 References

E.F.F. Botta and A.E.P. Veldman (1982) On local relaxation methods and their application to convection-diffusion equations. *J. Comput. Phys.* 48, 127–149.

A. Harten, J.M. Hyman and P.D. Lax (1976) On finite difference approximations and entropy conditions for shocks. *Comm. Pure Appl. Math.* 29, 297–322.

C. Hirsch (1990) *Numerical Computation of Internal and External Flows, Volumes 1 and 2.* John Wiley.

R.A. Horn and C.R. Johnson (1991) *Topics in Matrix Analysis.* Cambridge University Press.

A.S. Householder (1964) *The Theory of Matrices in Numerical Analysis.* Blaisdell Publishing Company (reprinted in 2006, Dover).

J.D. Lambert (1973) *Computational Methods in Ordinary Differential Equations.* John Wiley.

R.D. Richtmyer and K.W. Morton (1967) *Difference Methods for Initial Value Problems.* John Wiley.

A.E.P. Veldman (1996) *Partiele Differentiaalvergelijkingen.* Lecture notes (in Dutch), University of Groningen.

E.L. Wachspress (1966) *Iterative Solution of Elliptic Systems.* Prentice Hall.

D.M. Young (1971) *Iterative Solution of Large Linear Systems.* Academic Press.

Appendix B

Computer exercises

During the course several computer exercises have to be made. These exercises are to be solved in 'pairs' of one or two students. A (short) written report of the findings has to be handed in the following week; discussion of the homework with other 'pairs' of CFD students is not allowed during that period.

All software required for the exercises is available by downloading the files from the Nestor pages for this course:

`Computational Fluid Dynamics > Course Information`

An alternative is to download them through

`http://www.math.rug.nl/~veldman/Colleges/CFD/CFD-Exercises.html`

B.1 Exercise 1 – Artificial diffusion

Files and commands Required files: `cfd_1.m, cfd_upw.m, ns_exact.m`
The program is started in Matlab with the command `cfd_1`.

Description

Consider the inhomogeneous convection-diffusion equation

$$\frac{\mathrm{d}\phi}{\mathrm{d}x} - k\frac{\mathrm{d}^2\phi}{\mathrm{d}x^2} = S \quad \text{on} \quad 0 \le x \le 1, \quad \text{with} \quad \phi(0) = 0, \quad \phi(1) = 1.$$

The right-hand side is given by

$$S(x) = \begin{cases} 2 - 5x & , \quad 0 \le x \le 0.4, \\ 0 & , \quad 0.4 < x \le 1. \end{cases}$$

In this exercise an upwind discretization of the above equation is applied. The discrete solution for three values of the diffusion coefficient k is plotted in one picture. Also the exact solution for one value of k can be included in this picture.

Questions

1. Firstly, study the discrete solution for a grid with $N = 20$ grid points, at the following values of k: $k = 0.1$, 0.01 and 0.001. These values correspond with mesh-Péclet numbers $P = 0.5$, 5 and 50, respectively. One would expect (or at least hope) that these solutions tend to the (analytical) solution for $k = 0$. Hence, compare the discrete solutions to the exact solution at $k = 0$ (ignore the Matlab warnings concerning division by zero). What do you observe; is there any such relation?

2. Next, compare the same three discrete solutions at $k = 0.1$, 0.01 and 0.001 (again for $N = 20$) to the exact solution at $k = 0.025$. Now, there seems to be some relation between the upwind limit for $k \to 0$ and the exact solution at $k = 0.025$. Explain this relation.

3. Finally, study once more the upwind solutions at $k = 0.1$, 0.01 and 0.001, but now on a grid with $N = 40$ grid points. Which exact solution, i.e. at which value of k, corresponds with the upwind limit for $k \to 0$?

B.2 Exercise 2 – Various discretization methods

Description

As in Exercise 1, we consider the inhomogeneous convection-diffusion equation

$$\frac{\mathrm{d}\phi}{\mathrm{d}x} - k\frac{\mathrm{d}^2\phi}{\mathrm{d}x^2} = S \quad \text{on} \quad 0 \le x \le 1, \quad \text{with} \quad \phi(0) = 0, \quad \phi(1) = 1.$$

The right-hand side is given by

$$S(x) = \begin{cases} 2 - 5x & , \quad 0 \le x \le 0.4, \\ 0 & , \quad 0.4 < x \le 1. \end{cases}$$

The Matlab file `cfd_2` solves this equation with a number of finite-difference (-volume) methods. Several types of discretization are employed; the solutions obtained with these discretizations are shown in one combination figure – the exact solution is also indicated. Each figure shows eight smaller pictures, arranged as indicated below:

uniform grid	upwind	smart upwind	
lambda schemes	B3 ($\lambda = 1/2$)	QUICK ($\lambda = 1/8$)	central ($\lambda = 0$)
nonuniform grid	upwind	central A	central B

Two types of grid are used: uniform grids and non-uniform (stretched) grids. The stretched grid possesses $N/2$ equal grid cells between 0 and $1 - 5k$, and $N/2$ equal grid cells between $1 - 5k$ and 1.

The Matlab script also gives the eigenvalues of Method B; they are displayed in Matlab's main window.

Required files

This exercise requires the files `cfd_2.m, cfd_upw.m, cfd_lam.m, cfd_cen.m, ns_exact.m`.

Questions to be solved on the computer

Compute the solutions for $k = 0.05$, $k = 0.01$ and $k = 0.001$ (i.e. $P = 1$, 5 and 50). Use a grid with $N = 20$ grid cells

1. Compare the upwind results with those of the 'smart upwind' method for the same values of k; give a personal opinion of the usefulness of the latter method in comparison with the 'standard' upwind.

2. Compare the results of the lambda-schemes (cf. Section 1.3.2) with those of the upwind method; watch features like wiggle-dependency and artificial diffusion. Determine empirically for which value of k the QUICK method starts to wiggle, and explain this 'wiggle boundary' theoretically (see Question 5 below).

3. Finally, consider the results on the stretched grid. Compare the upwind method and both generalizations (A and B) of the central method (cf. Section 1.6). Describe the difference between the discrete solutions; why has the upwind solution not improved?

4. On the stretched grid, determine empirically the value of k for which one of the eigenvalues of the coefficient matrix of Method B crosses the imaginary axis. Furthermore, try to find a value of k for which the coefficient matrix is approximately singular. Watch how the solution of Method B behaves at this singularity.

Questions to be solved by pencil-and-paper

5. Prove that $\lambda \geq \max(\frac{1}{2} - \frac{k}{uh}, 0)$ is a sufficient condition for the upwind-biased lambda schemes from Eq. (1.15) on page 12 to be wiggle-free. In particular show that the QUICK method is wiggle-free for $P \leq 8/3$. Hint: Try fundamental solutions of the form r^i, and monitor the sign of r.

6. Discretize the convection-diffusion equation

$$\frac{\mathrm{d}\phi}{\mathrm{d}x} - k\frac{\mathrm{d}^2\phi}{\mathrm{d}x^2} = 0 \quad \text{on} \quad 0 \leq x \leq 1,$$

with Dirichlet boundary conditions

$$\phi(0) = 0, \quad \phi(1) = 1.$$

Use a nonuniform grid with only one interior grid point at $x = 1 - 2k$; k remains an unspecified variable here. Use the discretization methods A and B. In both cases, A and B, solve the discrete system (which in fact is only one equation). Sketch both solutions using linear interpolation between the grid points. Compute the exact solution of the convection-diffusion equation (at $x = 1 - 2k$) as well. Which discrete solution is better?

B.3 Exercise 3 – The JOR and SOR method

Description

After discretization of a (system of) partial differential equation(s) one obtains a large system of algebraic equations, $Ax = b$. Solving such a large system with a direct method is expensive with respect to both memory and CPU time. Therefore, such systems are often solved with an iterative method.

As in Exercise 1 and 2, we consider the inhomogeneous convection-diffusion equation

$$\frac{\mathrm{d}\phi}{\mathrm{d}x} - k\frac{\mathrm{d}^2\phi}{\mathrm{d}x^2} = S \quad \text{on} \quad 0 \leq x \leq 1, \quad \text{with} \quad \phi(0) = 0, \quad \phi(1) = 1.$$

The right-hand side is given by

$$S(x) = \begin{cases} 2 - 5x & , \quad 0 \leq x \leq 0.4, \\ 0 & , \quad 0.4 < x \leq 1. \end{cases}$$

The Matlab file `cfd_3.m` discretizes this equation with a number of finite-difference (-volume) methods, and solves the resulting linear systems iteratively.

Two types of grid are used: uniform grids and non-uniform (stretched) grids. The stretched grid possesses $N/2$ equal grid cells between 0 and $1 - 5k$, and $N/2$ equal grid cells between $1 - 5k$ and 1. Several types of discretization are employed; on the uniform grid both an upwind and a central discretization, and on the stretched grid an upwind, a central A and a central B discretization are used.

The resulting linear systems are solved iteratively with the JOR and SOR methods. As stopping criterion in these methods

$$\| \phi^{n+1} - \phi^n \|_\infty \leq 10^{-5}$$

is used, in which n is the counter of the iterative steps; the maximum norm of the difference between the solutions of two consecutive iteration steps must be sufficiently small.

Required files

This exercise requires the files `cfd_3.m, plotev.m, plotsol.m, ns_iterative.m, ns_jor.m, ns_sor.m, ns_exact.m`.

Typing the command `cfd_3` in Matlab starts a menu in which the choice of the grid, the discretization method, the iterative method and the value of the relaxation factor ω can be given. Then, with the menu a program can be started which solves the convection-diffusion equation in the prescribed way. The results produced by the program consist of the eigenvalues of the Jacobi matrix, a plot of the iteration error (this error is written to the screen as well), the total number of iterations and a plot showing the continuous solution, the discrete solution computed with a direct solver and the discrete solution computed with the iterative method (provided the iterative method did converge).

Questions to be solved on the computer

Compute the solution for $k = 0.01$. As a default, the program uses $N = 10$ grid cells.

1. Select the uniform grid; use first the upwind and then the central method. Determine numerically whether the Jacobi method (i.e. the JOR method with $\omega = 1$) converges or not. Do the same for the Gauss-Seidel method (i.e. the SOR method with $\omega = 1$). Put the results in the table below.

uniform grid	Jacobi method		Gauss-Seidel method	
	convergent?	# iterations	convergent?	# iterations
upwind				
central				

Use the eigenvalues of the Jacobi matrix, computed by `cfd_3`, to explain the results. When both methods converge, compare the convergence rate of the Jacobi and Gauss-Seidel method.

2. Select the uniform grid; use first the upwind and then the central method. Use various (suitably chosen) values of ω and determine numerically for each value of ω the number of iterations needed by the JOR method. Determine in this way the optimal and maximal relaxation factor, ω_{opt} and ω_{max}, for the JOR method. Do the same for the SOR method. Put the results in the table below.

uniform grid	JOR method				SOR method			
	ω_{opt}	# it.	ω_{max}	# it.	ω_{opt}	# it.	ω_{max}	# it.
upwind								
central								

Which systems are solved better by the JOR and SOR method, those resulting from the upwind discretization or those resulting from the central discretization?

3. Select the stretched grid; use first the upwind, then the central A and finally the central B method. Determine numerically whether the Jacobi method ($\omega = 1$) converges or not. Do the same for the Gauss-Seidel method ($\omega = 1$). Put the results in the table below.

stretched grid	Jacobi method		Gauss-Seidel method	
	convergent?	# iterations	convergent?	# iterations
upwind				
central A				
central B				

Use the eigenvalues of the Jacobi matrix, computed by `cfd_3`, to explain the results. When both methods converge, compare the convergence rate of the Jacobi and Gauss-Seidel method.

4. Select the stretched grid: use first the upwind, then the central A and finally the central B method. Use various (suitably chosen) values of ω and determine numerically for each value of ω the number of iterations needed by the JOR method. Determine in this way the optimal and maximal relaxation factor, ω_{opt} and ω_{max} for the JOR method. Do the same for the SOR method. Put the results in the table below.

stretched grid	JOR method				SOR method			
	ω_{opt}	# it.	ω_{max}	# it.	ω_{opt}	# it.	ω_{max}	# it.
upwind								
central A								
central B								

In case of the central B discretization compare the discrete solution obtained with the JOR or SOR method with the discrete solution obtained with the exact solver. Do the JOR and SOR method really converge? Use the eigenvalues of the Jacobi matrix, computed by `cfd_3`, to explain this result. Which systems are solved better by the JOR and SOR method, those resulting from the upwind discretization, those resulting from the central A discretization or those resulting from the central B discretization?

Questions to be solved by pencil-and-paper

5. In Section A.3.1 for a central discretization on the uniform grid the Jacobi matrix and its eigenvalues are derived. The Jacobi matrix reads

$$\text{diag}[\frac{P}{4}+\frac{1}{2}, 0, -\frac{P}{4}+\frac{1}{2}],$$

(P is the mesh-Péclet number) and its eigenvalues μ_J are given by

$$\mu_J = \frac{1}{2}\sqrt{4-P^2}\cos(\frac{i\pi}{N}), \quad i = 1, \ldots, N-1,$$

where N is the number of grid cells.
Derive in a similar way for the upwind discretization on the uniform grid the eigenvalues μ_J of the Jacobi matrix. Show that these eigenvalues are real and between -1 and 1 for every value of P.
Hint: the eigenvalues of the tri-diagonal matrix $\text{diag}(a, 0, c)$ are given by
$\mu = 2\sqrt{ac}\cos(i\pi/N), \quad i = 1, \ldots, N-1.$

6. For a central discretization on the uniform grid the eigenvalues of the Jacobi matrix are pure imaginary when $P > 2$. Let the largest pure imaginary eigenvalue be given by $\mu_J = i\mu_I$, with μ_I real. The optimal and maximal relaxation factor for the JOR method are given by (cf. Section A.3.1)

$$\omega_{\text{JOR,opt}} = \frac{1}{1+\mu_I^2}, \quad \omega_{\text{JOR,max}} = \frac{2}{1+\mu_I^2}.$$

For an upwind discretization on the uniform grid the eigenvalues of the Jacobi matrix are real (see Question 5). Let the largest real eigenvalues be given by $\mu_J = \mu_R$ and $\mu_J = -\mu_R$, with μ_R real. Derive, as a function of μ_R, the optimal and maximal relaxation factor for the JOR method.

Hint: the eigenvalues μ_{JOR} of the iteration matrix of the JOR method are related to the eigenvalues μ_J of the Jacobi matrix by $\mu_{\text{JOR}} = 1 + \omega(\mu_J - 1)$, and the JOR method converges when the eigenvalues μ_{JOR} lie within the unit circle.

Furthermore, for the SOR method give the optimal and maximal relaxation factor (as a function of μ_I and μ_R respectively) when using the central and the upwind discretization.

7. Use Questions 5 and 6 (remember that $k = 0.01$ and $N = 10$) to determine for the JOR method the values of the optimal and maximal relaxation factor when solving the systems resulting from the discretization on the uniform grid with respectively the upwind and central method. Do the same for the SOR method. Compare these theoretical results with the numerical results from Question 2.

8. Consider the upwind discretization on the stretched grid. Use Question 6 (remember that $k = 0.01$ and $N = 10$) and the numerical values of the eigenvalues μ_J of the Jacobi matrix, computed as a byproduct by `cfd_3` in Question 4, to determine for the JOR method the theoretical values of the optimal and maximal relaxation factor. Do the same for the SOR method. Compare these theoretical results with the numerical results for the upwind discretization on the stretched grid from Question 4.
Why is it difficult to compute the theoretical optimal and maximal relaxation factor when solving the systems resulting from a discretization on the stretched grid with a central A or central B method?

B.4 Exercise 4 – Time integration

Description

Consider the heat equation

$$\frac{\partial T}{\partial t} = \frac{\partial^2 T}{\partial x^2}, \quad 0 \le x \le 1, \quad 0 \le t \le 4,$$

with boundary conditions

$$T(0,t) = 0 \text{ and } \frac{\partial T}{\partial x}(1,t) = 0,$$

and initial condition

$$T(x,0) = x.$$

This equation is solved with a finite-difference method on a grid with N grid points ($N = 10$ or 20). The Neumann condition is discretized with a mirror point. The quantity $T(1,t)$ is monitored. As time-integration method the generalized Crank-Nicolson method is used. The discretized heat equation reads

$$\frac{T_j^{n+1} - T_j^n}{\delta t} = (1-\omega)\frac{T_{j+1}^n - 2T_j^n + T_{j-1}^n}{h^2} + \omega\frac{T_{j+1}^{n+1} - 2T_j^{n+1} + T_{j-1}^{n+1}}{h^2}.$$

The simulation program is available as Matlab file `cfd_4.m`.

Required files

This exercise requires the file `cfd_4.m`.

Typing the command `cfd_4` in Matlab asks for the input of the required parameters for this exercise: N, ω `(omega)` and δt.

Questions to be solved on the computer

1) First consider the fully explicit method ($\omega = 0$). Vary the number of grid points as $N = 10$ and $N = 20$. Solve the heat equation for various time steps, as indicated in the table below. Monitor whether the time integration is stable. In Question 4 you are asked to explain the stability limit on the time step.

time step	0.001 stable?	0.00125 stable?	0.00126 stable?	0.0025 stable?	0.005 stable?	0.0051 stable?	0.01 stable?
N=10							
N=20							

2) Next, use the Crank-Nicolson method ($\omega = 0.5$) and the generalized Crank-Nicolson method with $\omega = 0.6$. Use $N = 10$. Set the time step δt at 0.05 and 0.5; see the table below. For the larger time step the solution shows clear oscillations. Make a rough estimate for the damping factor (defined as the quotient of two successive amplitudes) of the oscillations.

time step	$\delta t = 0.05$	$\delta t = 0.5$	
	stable?	stable?	damping
$\omega = 0.5$			
$\omega = 0.6$			

3) Finally, again for $N = 10$, investigate the fully implicit method ($\omega = 1$). Use the same time steps as in Question 2. Observe whether or not the solution shows oscillations.

time step	$\delta t = 0.05$		$\delta t = 0.5$	
	stable?	oscillations?	stable?	oscillations?
$\omega = 1.0$				

Questions to be solved by pencil-and-paper

4) First consider the fully explicit method ($\omega = 0$). Carry out a Fourier analysis of this method. Determine the maximum allowable time step for $N = 10$ and $N = 20$. Compare these stability limits with the empirical observations in Question 1.

5) Carry out a Fourier stability analysis of the generalized Crank-Nicolson method, and determine the amplification factor. Show that the generalized Crank-Nicolson method is unconditionally stable for $\omega \geq 0.5$. Investigate how the amplification factor behaves

when $\delta t \to \infty$. Now explain the oscillations visible in Question 2. How are these oscillations influenced when ω is increased from 0.5 to 0.6? Give a possible explanation why the observed amplification factors in Question 2 are somewhat different from the theoretical amplification factors computed in this Question.

6) Derive theoretically for which values of the Crank-Nicolson parameter ω the solution is wiggle-free. As a special case, explain why the fully implicit method ($\omega = 1$) does not show oscillations (see Question 3). Hint: Use the concept of a positive operator.

B.5 Exercise 5 – Navier–Stokes solver

General description

After semi-discretization with an explicit time-discretization method with two time levels, the Navier–Stokes equations can be written as

$$\begin{aligned} \text{div } \boldsymbol{u}^{n+1} &= 0, \\ \frac{\boldsymbol{u}^{n+1} - \boldsymbol{u}^n}{\delta t} + \text{grad } p^{n+1} &= \boldsymbol{R}^n, \end{aligned}$$

in which $\boldsymbol{R} = -(\boldsymbol{u}.\text{grad})\boldsymbol{u} + \nu \text{ div grad } \boldsymbol{u}$ consists of the convection and diffusion terms. Rewriting and combining these two equations gives

$$\text{div grad } p^{n+1} = \text{div } (\frac{\boldsymbol{u}^n}{\delta t} + \boldsymbol{R}^n), \tag{1a}$$

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^n + \delta t \ \boldsymbol{R}^n - \delta t \text{ grad } p^{n+1}. \tag{1b}$$

The fortran program `cfd_5.f` solves the Navier–Stokes equations.

Files and commands

This exercise requires the files `cfd_5.f, cfd.in, liqdmenu.m, liqdplot.m, liqprint.m, xplot.m`.

The program `cfd_5.f` has to be compiled with the command `gfortran -o cfd_5 -O cfd_5.f` (any other Fortran compiler, e.g. `ifort`, will also do). Then, the program can be run with the command `./cfd_5`. This program needs the input file `cfd.in`, this file has to be adjusted to perform the different computations. The input file, with the parameter setting of Question 0, is included at the end of this exercise. At the end of the input file an explanation of the different parameters is given.

The program writes output both to the screen and to files. The output to the screen first gives the values of the parameters describing the problem, and then gives each (user specified) time interval the time, div $\boldsymbol{u}$, the number of Poisson iterations and the horizontal velocities at three positions at the vertical center line of the domain (at the bottom, halfway, and at the top). Furthermore, output is written to the files `uvpf#.dat` and `config.dat`. The output file `config.dat` contains a description of the grid. The output file `uvpf#.dat` contains the horizontal and vertical velocity and pressure in each grid cell at a certain time. In the input file the user can specify whether the file `uvpf#.dat` is made each (user specified) time interval

or only at the end of the computation.

The output files `uvpf#.dat` and `config.dat` are needed for the post-processing in Matlab. Typing the command `liqdmenu` in Matlab starts a menu with which the post-processing can be controlled. With the upper part of the menu surface or contour plots of the (different) velocities, pressure, vorticity or streamfunction can be made and with the lower part of the menu a movie of these plots at consecutive time steps can be made. With the middle part of the menu plots of the (different) velocities or the pressure along horizontal or vertical sections of the domain can be made.

Part I: Treatment of div $\boldsymbol{u}^n$

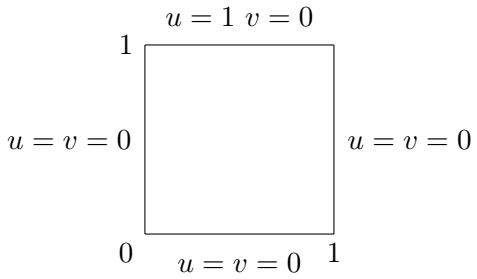
Description

The term div $\boldsymbol{u}^n = 0$ in the semi-discretized Navier–Stokes equations can be treated in two ways. The first way is to maintain this term in the equations. In this case equations (1a) and (1b) have to be solved. However, when the Poisson equation is solved exactly, div $\boldsymbol{u}^{n+1} = 0$. Therefore, the second way is to substitute this in the right-hand side of equation (1a) in the next time step (i.e. substitute div $\boldsymbol{u}^n = 0$). In this case the following equations have to be solved

$$\text{div grad } p^{n+1} = \text{div } \boldsymbol{R}^n, \tag{2a}$$

$$\boldsymbol{u}^{n+1} = \boldsymbol{u}^n + \delta t\ \boldsymbol{R}^n - \delta t \text{ grad } p^{n+1}. \tag{2b}$$

Consider the lid-driven cavity problem; the flow in a square cavity with constantly moving lid. This problem is a well known test case for numerical methods for solving the Navier–Stokes equations. The domain and boundary conditions of this problem are shown in the figure below. We will solve the lid-driven cavity problem with $\nu = 0.01$.



Parameter setting in the input file

Since we solve the lid-driven cavity problem with $\nu = 0.01$, set `Nu=0.01` in the input file. Use a stretched grid with 16 cells in each direction, i.e. set `iMaxUs=jMaxUs=18`. Take smaller cells at the boundaries of the cavity, set `xpos=ypos=0.5` and stretching parameters `cx=cy=-0.5`. Discretize the convection terms with a central method, i.e. set `Alpha=0.0`.

Since the velocity at the top is prescribed, set `top=8`, `Uin=1.0` and `Vin=0.0` in the input file. Furthermore, since at the other boundaries we have no-slip conditions, set `left=2`,

`right=2` and `bottom=2`.

Since the program was originally developed to compute non-stationary flows, it does not contain a stopping criterion which checks whether a stationary solution is reached or not. Instead, the program stops when the simulation time is equal to a user prescribed time (TFin), take `TFin=25`. Set the time step at `DelT=0.025` and the maximal allowed CFL number at `CFLMax=0.5`. The program automatically adjusts the time step such that the CFL number is between 0.4*CFLMax and CFLMax. In each time step the Poisson equation is solved with MILU. Use in the input file `PrtDt=1` and `uvp=0`, i.e. every second output is written to the screen and the velocity field is written to the file `uvpf0001.dat` at the end of the simulation.

Question 0: 'exact' solution

First the 'exact' solution is computed, i.e. the discrete steady solution on the given grid. Hereto set in the input file `Divuv=1` (i.e. retain the term div $\boldsymbol{u}^n$ in the Poisson equation), `StrtP=1` (i.e. use the previously found pressure as initial guess for the iterations in the new time step) and $\varepsilon = 10^{-3}$. Note that the solution has converged in time, and write the results at $t = 25$ in the table below. Note that this table directly matches the output of the program.

time	div $\boldsymbol{u}$	# it.	u bottom	u mid	u top
25					

Questions without div $\boldsymbol{u}^n$

First, substitute div $\boldsymbol{u}^n = 0$ in the Poisson equation (i.e. set `Divuv=0` in the input file). Hence, solve the Navier–Stokes equations in the form (2a) and (2b). In each of the following questions, put the output of the program at $t = 20, 21, \ldots, 25$ in a table of the format given below.

time	div $\boldsymbol{u}$	# it.	u bottom	u mid	u top
20					
21					
22					
23					
24					
25					

1) Use for the Poisson solver an accuracy $\varepsilon = 10^{-1}$ and use as initial solution for the Poisson solver $p = 0$, i.e. start the Poisson solver from scratch.

   a) Look what happens with the term div $\boldsymbol{u}$ in time. Furthermore, make a plot of the solution; e.g. use the post-processing menu in Matlab to view the flow, use the upper part of the menu to make a surface plot of the streamfunction. Also have a look at the absolute velocity (`crange` [0, 0.1]) and the vertical velocity. Repeat the simulation with `TFin=100`. Does the velocity field converge?

b) Now we will consider what happens with the divergence when $\delta t \to 0$: will this improve the solution? Set (temporarily) `DelT=0.0025` and `CFLMax=0.05` in the input file. Repeat the simulation (with `TFin=25`). Look what happens with the term div $\boldsymbol{u}$ in time. Compare this with the behaviour of div $\boldsymbol{u}$ in Question 1a. Explain the results.

2) Use for the Poisson solver an accuracy $\varepsilon = 10^{-3}$, and use as initial solution for the Poisson solver $p = 0$, i.e. start the Poisson solver from scratch. Look what happens with the term div $\boldsymbol{u}$ in time. Furthermore, make various plots of the solution as in the previous question. The solution has improved, but is it really accurate?

3) Use for the Poisson solver an accuracy $\varepsilon = 10^{-3}$, and use as initial solution for the Poisson solver the solution from the previous time step. Look once more what happens with the term div $\boldsymbol{u}$ in time. Are you satisfied now?

4) Use for the Poisson solver an accuracy $\varepsilon = 10^{-1}$, and use as initial solution for the Poisson solver the solution from the previous time step. Once again, describe the behaviour of div $\boldsymbol{u}$ in time.

Questions with div $\boldsymbol{u}^n$

Next, retain the term div $\boldsymbol{u}^n$ in the Poisson equation (i.e. set `Divuv=1` in the input file). Hence, solve the Navier–Stokes equations in the form (1a) and (1b).

5) Use for the Poisson solver an accuracy $\varepsilon = 10^{-1}$ and use as initial solution for the Poisson solver $p = 0$, i.e. start the Poisson solver from scratch.

   a) Look what happens with the term div $\boldsymbol{u}$ in time. How do you like the solution?

   b) Now we will consider what happens with the divergence when we let $\delta t \to 0$. Set (temporarily) `DelT=0.0025` and `CFLMax=0.05` in the input file. Repeat the simulation. Look what happens with the term div $\boldsymbol{u}$ in time. Compare the results with those from Question 5a, and explain the differences.

6) Use for the Poisson solver an accuracy $\varepsilon = 10^{-3}$, and use as initial solution for the Poisson solver $p = 0$, i.e. start the Poisson solver from scratch (set `StrtP=0` in the input file). Look what happens with the term div $\boldsymbol{u}$ in time. Compare the results with those from Question 5a, and explain the differences.

7) Use for the Poisson solver an accuracy $\varepsilon = 10^{-3}$, and use as initial solution for the Poisson solver the solution from the previous time step (i.e. set `StrtP=1` in the input file). Monitor div $\boldsymbol{u}$ in time, and make plots of the solution. Explain the results.

8) Use for the Poisson solver an accuracy $\varepsilon = 10^{-1}$, and use as initial solution for the Poisson solver the solution from the previous time step. Monitor div $\boldsymbol{u}$ and plot the solution. How do you like the results?

Summary We have seen that the various approaches give quite different results. In the last question of this part you will have to decide which approaches give acceptable results.

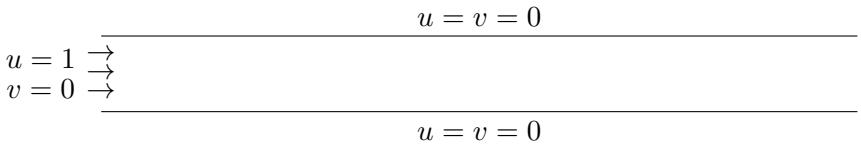
9) In practical applications one is often satisfied when the final solution has an accuracy of two digits. Keeping this in mind, compare the results from the Questions 1 to 8 and give your opinion about which approaches give acceptable results?

Part II: Influence of boundary conditions

Description

We will now consider the choice of boundary conditions at in- and outflow openings. The inflow boundary usually does not lead to too many problems. However, the outflow boundary is much more problematic. When at an outflow boundary the normal velocity is prescribed then a boundary layer may be created.

Consider the flow in a channel of length 10 and height 1. At the left (inflow) boundary the fluid flows into the channel with a horizontal velocity $u = 1$. At the upper and lower boundary of the channel we impose no-slip conditions. The geometry and (part of) the boundary conditions are given in the figure below.



We will use two types of boundary conditions at the right (outflow) boundary of the channel. Firstly, we will prescribe a horizontal velocity $u = 1$ at the right (outflow) boundary of the channel as well. Secondly, we will allow a free outflow, obtained by setting the pressure at 0, at the right (outflow) boundary. We will consider the influence of these boundary conditions on the solution.

Parameter setting in the input file

Use a grid without stretching with 32 cells in x-direction and 17 cells in y-direction, i.e. use `iMaxUs=34` and `jMaxUs=19` and `cx=cy=0.0` in the input file. Note that the length of the channel is 10, i.e. `Xmax=10`. Discretize the convection terms with a central method, i.e. set `Alpha=0.0`.

At the left (inflow) boundary the velocity is prescribed, hence set `left=8`, `Uin=1`, and `Vin=0`. Furthermore, at the upper and lower boundary no-slip conditions are imposed, i.e. set `top=2` and `bottom=2`.

Maintain the term div $\boldsymbol{u}^n$ in the Poisson equation (i.e. solve the Navier–Stokes equations in the form (1a) and (1b), set `Divuv=1` in the input file). Solve the Poisson equation with an accuracy $\varepsilon = 10^{-4}$. The Poisson equation is solved with MILU. Use as initial solution for the Poisson solver the solution from the previous time step (i.e. set `StrtP=1`).

Simulate the flow in the channel for 15 seconds, that is set `TFin=15` in the input file. Set the time step at `DelT=0.025` and the maximal allowed CFL number at `CFLMax=0.1`. Use in

the input file `PrtDt=1` and `uvp=1`, i.e. every second of the simulation output is written to the screen and every second of the simulation the velocity field is written to a file `uvpf#.dat` (be aware: this takes a lot of memory).

In every question below, plot the flow at the first five seconds of the simulation and at the end of the simulation, i.e. at $t = 1, 2, \ldots, 5$ and $t = 15$. Use the post-processing menu in Matlab to plot the flow. Use the upper part of the menu to make a surface plot of the absolute velocity. Furthermore, use the middle part of the menu to make a plot of the horizontal velocity along the horizontal center line (i.e. j-plane=10). By using the '`hold`' button in the menu the plots of the horizontal velocity along the horizontal center line at $t = 1, 2, \ldots, 5$ and $t = 15$ can be put in one figure. A boundary layer and irregularities of the flow can easily be seen in the second plot.

Questions with prescribed outflow

Firstly, prescribe a horizontal velocity at the right (outflow) boundary of the channel as well: $u = 1$ and $v = 0$ (i.e. set `right=8` in the input file).

10) Simulate the flow for $\nu = 0.01$. Make plots of the flow as described above. Describe and explain the development in time of the horizontal velocity along the horizontal center line.

11) Simulate the flow for $\nu = 0.005$. Make plots of the flow as described above. Compare the flow with the one computed in Question 10, and explain the differences.

12) Simulate the flow for $\nu = 0.001$. Make plots of the flow as described above. Compare the flow with those computed in Question 10 and 11, and explain the differences. Furthermore, make a surface plot of the absolute velocity in which the vector plot is included (select in the menu '`arrows automatic`' instead of '`no velocity vectors`'), and zoom first in on the x-interval [0,1] and then on the x-interval [9,10] (i.e. take '`manual x-axis`' instead of '`automatic zoom`' and select the interval). Furthermore, make a plot of the horizontal velocity at the vertical section at the i-plane 32. What happens with the velocity at the upper and lower boundary?

Questions with free outflow

Secondly, allow at the right boundary of the channel a free outflow condition (i.e. set `right=7` in the input file).

13) Simulate the flow for $\nu = 0.01$. Make plots of the flow as described above. Compare the flow with the one computed in Question 10, and explain the differences. What is the limit value of the horizontal velocity at the center line $y = 0.5$.
When the flow is fully developed the (horizontal) velocity profile is a parabola. Compute this parabola theoretically. What is the theoretical value of the horizontal velocity of the fully developed flow at the center line $y = 0.5$. Compare this result with the numerical value. Also make a cross sectional plot of the horizontal velocity near the end of the channel to check the parabolic profile.

14) Simulate the flow for $\nu = 0.005$. Make plots of the flow as described above. Compare the flow with those computed in Question 11 and 13, and explain the differences. What

is the limit value of the horizontal velocity at the center line $y = 0.5$. Compare this numerical value with the theoretical value computed in Question 13.

15) Simulate the flow for $\nu = 0.001$. Make plots of the flow as described above. Compare the flow with those computed in Question 12, 13 and 14 and explain the differences. What is the value of the horizontal velocity at the center line $y = 0.5$ at the end of the channel. Explain why this numerical value of the horizontal velocity at the center line $y = 0.5$ differs from the theoretical value computed in Question 13. Also make a cross sectional plot of the horizontal velocity near the end of the channel to see the shape of the velocity profile: is it a parabola?

Repeat the simulation with a channel of respectively length 20 and length 50 (set first `Xmax=20` and then `Xmax=50`), use `TFin=50` in order to reach a steady-state solution; also increase `Imax` (although it has little influence). Set `uvp=0` in order to save memory. Make only at the end of the simulation (i.e. at $t = 50$) a plot of the horizontal velocity along the horizontal center line (i.e. j-plane=10). What is the limit value of the horizontal velocity at the center line $y = 0.5$. Again make a cross sectional plot of the horizontal velocity near the end of the channel to see the shape of the velocity profile: is it a parabola now? Find a physical explanation of the results.

Example input file

```
***** tank geometry ************************************************
Xmin      Xmax     Ymin     Ymax
0.0       1.0      0.0      1.0

***** grid definition **********************************************
iMaxUs   jMaxUs    cx    cy     xpos    ypos
  18      18     -0.5   -0.5    0.5      0.5

***** liquid properties ********************************************
 Nu
0.01

***** boundary conditions and inflow characteristics ***************
left    right   top   bottom   UIn   VIn
 2        2      8       2     1.0   0.0

***** discretization ***********************************************
Alpha   Divuv
0.0      1

***** poisson iteration parameters *********************************
Epsi      ItMax  StrtP
1.0e-4     100     1

***** time step  ***************************************************
TFin   DelT    CFLMax
25    0.025    0.5

***** print/plot control *******************************************
```

```
PrtDt  uvp
1.0     0

------------------------------------------------------------------------
***** E X P L A N A T I O N ************ E X P L A N A T I O N *****
------------------------------------------------------------------------

**TANK GEOMETRY**
Xmin, Xmax, Ymin, Ymax : position of boundaries of computational domain

**GRID DEFINITION**
iMaxUs (<=130), jMaxUs (<=130) : number of cells including mirror cells,
                                 the number of 'real' cells is iMaxUs-2, jMaxus-2
cx, cy     : stretchparameters - 0=no stretch;
                                 >0=smaller cells near position
                                    indicated by xpos,ypos;
                                 <0=smaller cells away from xpos,ypos

**LIQUID PROPERTIES**
Nu         : kinematic viscosity


**BOUNDARY CONDITIONS AND INFLOW CHARACTERISTICS**
left,right,top,bottom  : type of boundary condition
                             1=slip
                             2=no-slip
                             7=free outflow : as boundary condition the pressure
                                              is set at 0
                             8=inflow  : the velocities are prescribed
UIn,VIn    : in case of inflow conditions, prescribed velocities
             the sign of UIn/VIn corresponds to the x/y-direction (hence it
             does not necessarily correspond to the inward pointing direction)

**DISCRETIZATION**
Alpha      : upwind parameter, discretization of convection term
                0 = central
                1 = upwind
Divuv      : treatment of divergence
                0 = substitute div u^n=0 in Poisson equation
                1 = maintain div u^n in Poisson equation

**POISSON ITERATION PARAMETERS**
Epsi       : convergence criterion of Poisson solver
ItMax      : maximal allowed number of iterations of Poisson solver
Strtp      : initial field of Poisson solver
               0 = zero field p=0
               1 = solution from previous time step, p=poud;

**TIME STEP**
TFin       : endtime of simulation
DelT       : timestep (is adjusted during the simulation in order to
             keep the CFL number smaller than CFLMax)
CFLMax     : maximum allowed CFL number
```

```
** PRINT/PLOT CONTROL**
PrtDt      : time between 2 small printouts (to the screen and a file)
uvp        : velocity and pressure in Matlab format
               0=only at end of computation (preferable)
               1=at every small printout (takes a lot of memory)
```