Simulation of flow in a glass tank

Simulation of Flow in a Glass Tank

V.Nefedov and R.M.M.Mattheij

Scientific Computing group
Department of Mathematics and Computing Science
Eindhoven University of Technology, PO BOX 513
5600MB Eindhoven, The Netherlands
email: nefedov@win.tue.nl, mattheij@win.tue.nl

Abstract

This paper describes a model to perform simulation of the melting process of glass in an oven. The Navier-Stokes and energy equations are discretized by a finite volume method on a collocated grid. A special feature is that we employ refinement by uniform grids and solve the global system by so called local defect correction. The method is illustrated by two practicle examples.

1 Introduction

The manufacturing of glass is a complicated and expensive process. The glass is produced in a so-called glass oven or tank. The raw materials such as soda and sand are dumped on one side of the oven, which is heated from above by gas burners. Ovens are constructed in such a way that the glass stays inside for some time, and after about 20 hours flows via feeders to the production lines on the other side. There are, therefore, several processes involved, such as flow and heat transer and various chemical reactions. In order to study how different oven configurations affect the production, numerical simulation is required, as full scale experimental studies are too expensive to carry out.

Since the complexity of the problem is quite appalling, existing codes for tackling this problem are in for improvement and further sophistication, like

adaptivity in gridding, to make the simulations more feasible (with respect to memory and computing time). In [6] a local refinement procedure was combined with a glass oven model. The method proposed was based on staggered grid finite volumes [7]. In this paper we study a different way to simulate the glas flow, based on locally uniform grids. The next section describes the mathematical model, namely the equations we are dealing with, the boundary conditions and the specific physical parameters. In the third section we discuss the discretisation procedure. In section 4 the solution method is described, including some implementation details. Section 5 deals with a local refinement technique called local defect correction (LDC). Its application to the glass flow is discussed in the two last sections, viz. a stirred flow in section 6 and bubbling in section 7.

2 Mathematical model

The oven in fact consists of an oven proper (melting tank) and a second part, connected by a small channel, from which the glass is going to a feeder channel for further processing, leading to the actual product eventually, see Fig.1.

The mathematical model consists of equations, describing physical processes occuring in the glass flow, complemented by suitable boundary conditions. In order to restrict this still general model to our particular case of interest we perform a dimension analysis based on glass glass flow properties.

2.1 Main equations

Since melted glass can be considered a viscous Newtonian fluid, we can model it by the (incompressible) Navier-Stokes equations

$$\begin{cases} (\rho(\mathbf{x})\mathbf{u}, \nabla\mathbf{u}) = \mathbf{F} - \nabla p + \nabla \cdot (\mu(\mathbf{x})\nabla\mathbf{u}), \\ \nabla \cdot (\rho(\mathbf{x})\mathbf{u}) = 0. \end{cases} \tag{1}$$

Here μ, ρ are the viscosity and the density of the glass respectively, and $\mathbf{F} = (0, 0, -\rho g)^T$ is a gravitational force. Since the glass flow exibits steady state behaviour for large time scales, these equations are written in a time independent form.

The motion of the glass is caused by temperature differences; thus the set of equations is incomplete without the energy equation. For a steady low-

velocity flow with negligible dissipation we can derive the energy equation in the following form

$$\nabla \cdot (C_p \rho(\mathbf{x}) \mathbf{u} T) = \nabla \cdot (k(\mathbf{x}) \nabla T), \tag{2}$$

where k and C_p are the conductivity and the heat capacity, respectively.

2.2 Boundary conditions

A typical glass oven configuration is depicted in Figure 1. We can define

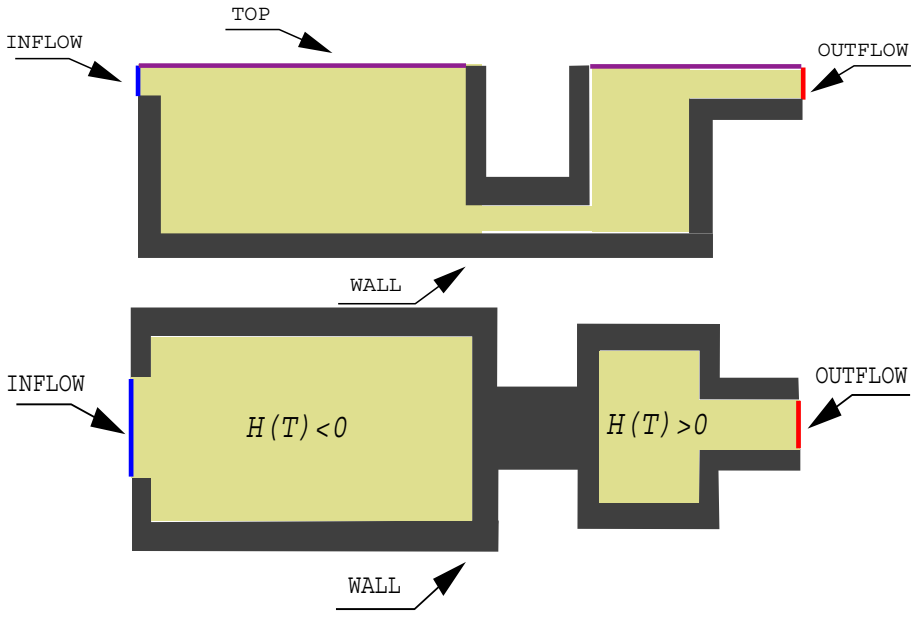

Figure 1: Sketch of the glass tank (horizontal and vertical cross sections)

the boundary values for this geometry as follows: the velocity has Dirichlet values on the inflow, outflow and, assuming no-slip, on the walls too. The top layer is modelled as a symmetry plane, that is the normal component is zero; for the rest we prescribe homogeneous Neumann boundary conditions.

For the temperature the situation is different, since the heat loss from the walls and heat influx from the top layer are not known a priori, but can be expressed via a Robin-type boundary condition. In particular we can write

$$-(\mathbf{u}, \mathbf{n})|_{\Gamma_{inflow}} = u_0,$$

$$
\begin{gathered}
-(\mathbf{u},\mathbf{n})|_{\Gamma_{outflow}} = u_1, \\
(\mathbf{u},\tau)|_{\Gamma_{inflow}} = (\mathbf{u},\tau)|_{\Gamma_{outflow}} = 0, \\
\mathbf{u}|_{\Gamma_{w,g}} = 0, \\
(\mathbf{u},\mathbf{n})|_{\Gamma_{top}} = 0, \quad \frac{\partial(\mathbf{u},\tau)}{\partial \mathbf{n}}|_{\Gamma_{top}} = 0, \\
T|_{\Gamma_{inflow}} = T_0, \quad \frac{\partial T}{\partial \mathbf{n}}|_{\Gamma_{outflow}} = 0, \\
\frac{\partial T}{\partial \mathbf{n}}|_{\Gamma_{top}} = H(T), \quad \frac{\partial T}{\partial \mathbf{n}}|_{\Gamma_{wall}} = Q(T).
\end{gathered}
$$

2.3 Physical parameters

The above described equations, together with the boundary conditions, are still too general, since they might as well represent any viscous flow in a tank. In order to resctrict the model to the particular case of the glass flow we are interested in, we need to specify the parameters of the equation, in particualr the viscosity, the density, the heat conductivity and the heat capacity.

The most important of these coefficients are the viscosity and the density. The viscosity does significantly affect the flow pattern; it decays exponentially as the temperature grows (Vogel Fucher Tamman law)

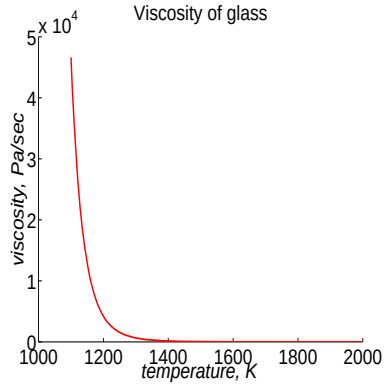

Figure 2: Viscosity of tv-glass

$$
\mu_{glass}(\mathbf{x}) = \mu_{glass}(T(\mathbf{x}) = a_\mu e^{b_\mu/(T-c_\mu)}. \tag{3}
$$

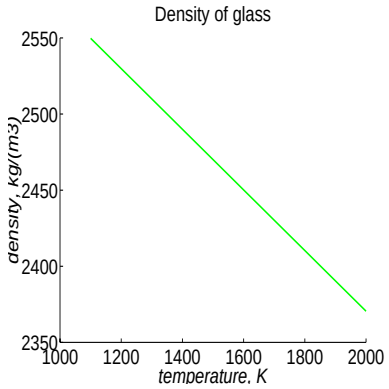

Figure 3: Density of tv-glass

The coefficients a_μ, b_μ, c_μ in (3) are specific for the glass type (tv-glass, window-glass etc).

The most significant factor in a glass flow computation is the density. It may be modelled as a linear function of the temperature, more precisely

$$\rho_{glass}(T) = a_\rho(1 - b_\rho(T - c_\rho)), \tag{4}$$

and changes by only about 10%; nevertheless it drives the flow via the convective term and the gravitation.

2.4 Dimensional analysis

In order to examine the behaviour of the momentum and the energy equations, in particular to determine whether diffusion or convection prevails, we make the equations dimensionless. Let us rewrite them as new variables

$$\tilde{\mathbf{x}} := \frac{1}{X}\mathbf{x}, \quad \tilde{\mathbf{u}} := \frac{1}{U}\mathbf{u}, \quad \tilde{T} := \frac{T - T_{min}}{\Delta T}.$$

Here X, U are characteristic length and velocity respectively and $\Delta T = T_{max} - T_{min}$ is a maximum possible temperature difference. The gradient and the divergence operators in the old and the new variables are related as

$$\tilde{\nabla} := \frac{1}{X}\nabla, \quad \tilde{\nabla}\cdot := \frac{1}{X}\cdot\nabla.$$

A tilde over the gradient and the divergence thus indicates that it is taken with respect to the new variables. After subsitution the Navier-Stokes and the energy equations will look like

$$\begin{cases} \dfrac{U^2}{X}(\rho\tilde{\mathbf{u}}, \tilde{\nabla}\tilde{\mathbf{u}}) = F - \dfrac{1}{X}\tilde{\nabla}p + \dfrac{U}{X^2}\tilde{\nabla}\cdot(\mu\tilde{\nabla}\tilde{\mathbf{u}}), \\ \tilde{\nabla}\cdot(\rho\tilde{\mathbf{u}}) = 0, \end{cases} \tag{5}$$

$$\frac{\Delta TU}{X}C_p(\rho\tilde{\mathbf{u}}, \tilde{\nabla}\tilde{T}) = \frac{\Delta T}{X^2}\tilde{\nabla}\cdot(k\tilde{\nabla}\tilde{T}). \tag{6}$$

Since the density ρ changes by only about 10% we assume ρ to be constant. After dividing both parts of the first equation of (5) by $\rho U^2/X$ we obtain an equation in dimensionless variables

$$(\tilde{\mathbf{u}}, \tilde{\nabla}\tilde{\mathbf{u}}) = \tilde{F} - \tilde{\nabla}\tilde{p} + \tilde{\nabla}\cdot(\frac{1}{Re}\tilde{\nabla}\tilde{\mathbf{u}}).$$

Here

$$\tilde{F} = \frac{X}{\rho U^2}F, \quad \tilde{p} = \frac{1}{\rho U^2}p.$$

By analogy we divide both parts of (6) by $\Delta TUC_p/X$ to obtain

$$(\tilde{\mathbf{u}}, \tilde{\nabla}\tilde{T}) = \tilde{\nabla}\cdot(\frac{1}{Re\cdot Pr}\tilde{\nabla}\tilde{T}).$$

Here the *Reynolds* (Re) and *Prandtl* (Pe) numbers are defined by

$$Re := \frac{\rho UX}{\mu}, \quad Pr := \frac{\mu C_p}{k}$$

The Reynolds number expresses the ratio between convection and diffusion, while the product of Reynolds and Prandtl numbers indicates correspondence between radiative and convective heat transfer. In the case of the glass tank (ommiting physical units)

$$X = 20,\ \ U = 0.01,\ \ \rho \sim 2490,\ \ 25 \le \mu \le 250,\ \ 1.2 \le k \le 64,\ \ C_p = 1000.$$

That is

$$0.2 \le Re \le 20, \quad 7812.5 \le RePr \le 416667.$$

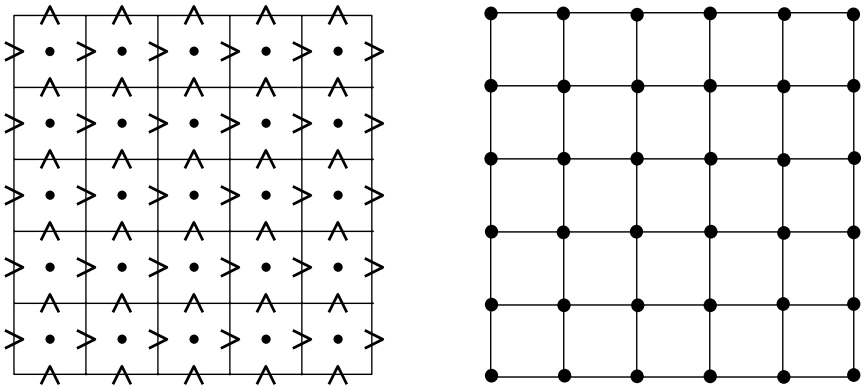

Figure 4: Staggered (left) and collocated (right) grids

3 Discretisation

The next step towards the solution is a discretisation of the continuous problem. We have opted for the finite volume method, because it ensures conservation of a number of important properties, such as momentum and mass. After the discretisation procedure has been fixed, we need to choose the grid on which we will discretise the equations.

3.1 Collocated grid versus staggered grid

The natural choice of the grid for a finite volume method is a *staggered grid* [7]. It is called staggered because the velocity components are staggered with respect to the pressure, which is placed in the center of the cells. The staggered grid ensures that the resulting discrete system will not be singular. In some sense it plays the same role as the LBB or Inf-Sup condition in the finite element method [8]. Despite the fact that using a staggered grid we always end up with a non-singular system, the programming process using such a grid is not a triviality, especially in 3D.

We will use another type of grid, namely a collocated grid, that is the grid where all variables are computed at the same locations. It is far more convenient for programming, but requires a special discretisation procedure to ensure non-singularity of the system.

3.2 Discretisation of the momentum equations

Let us consider the momentum equation corresponding to the horizontal component of the velocity

$$-\nabla \cdot (\mu(T)\nabla u) + (\rho \mathbf{u}, \nabla u) + \frac{\partial p}{\partial x} = 0. \tag{7}$$

For the sake of simplicity we consider here the two dimensional case. Momentum and body force are discretised according to a standard finite volume procedure [3]. Let us restrict our attention to the pressure gradient. Integrating the first component over a control volume, with node C say,

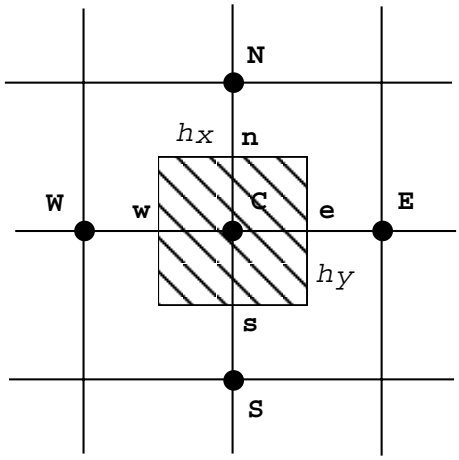

Figure 5: Pressure stencil

$$\int_{V_c} \frac{\partial p}{\partial x} dV \doteq h_y(p_e - p_w) \doteq h_y(p_E - p_C). \tag{8}$$

We use downstream values of the pressure in (8) instead of midvalues. It leads to first order approximation instead of a possibly second order but, as it will be explained later, this will guarantee the non-singularity of the system.

3.3 Discretisation of the continuity equation

Since we are using a collocated grid, the continuity equation is integrated over the same control volumes as the momentum equations.

$$\int\limits_{V_c} \left(\frac{\partial(\rho u)}{\partial x} + \frac{\partial(\rho v)}{\partial y} \right) dV \doteq h_y[(\rho u)_e - (\rho u)_w] + h_x[(\rho v)_n - (\rho v)_s] \doteq$$
$$h_y[(\rho u)_C - (\rho u)_W] + h_x[(\rho v)_C - (\rho v)_S]. \tag{9}$$

Here the upstream values of the velocity are used for the same reason as in the pressure gradient approximation.

4 PISO method

A solution method for the Navier-Stokes equations coupled with the energy equation was suggested by Issa [4]. It is an operator splitting approach, which is closely related to the well-known family of SIMPLE/SIMPLER algorithms by Patankar [7].

Let us assume now that after discretisation the system has the following form

$$\begin{pmatrix} \mathbf{M}(\mathbf{u},\mathbf{T}) & \mathbf{B^p} & \mathbf{0} \\ \mathbf{B^u}\rho(\mathbf{T}) & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{E}(\mathbf{u},\mathbf{T}) \end{pmatrix} \begin{pmatrix} \mathbf{u} \\ \mathbf{p} \\ \mathbf{T} \end{pmatrix} = \begin{pmatrix} \mathbf{F} \\ \mathbf{0} \\ \mathbf{0} \end{pmatrix}, \tag{10}$$

Here $\mathbf{M}(\mathbf{u},\mathbf{T})$ is a discrete momentum operator, $\mathbf{E}(\mathbf{u},\mathbf{T})$ — discrete energy operator, both depending non-linearly on the velocity and the temperature. $\mathbf{B^p}$ and $\mathbf{B^u}$ are the discrete gradient and the discrete divergence, respectively.

4.1 Description of the algorithm

First we append to the momentum equations a time dependent term. Iterants $\mathbf{u}^n$, $\mathbf{T}^n$ and $\mathbf{p}^n$ are assumed to be known. After solving the prediction step

$$\frac{\rho(\mathbf{T}^n)\mathbf{u}^* - \rho(\mathbf{T}^n)\mathbf{u}^n}{\Delta t} = \mathbf{F} - \mathbf{M}(\mathbf{u}^n,\mathbf{T}^n)\mathbf{u}^* - \mathbf{B^p}\mathbf{p}^n \tag{11}$$

we find the velocity field $\mathbf{u}^*$ which is our first guess. Since it might not satisfy the continuity equation we correct it in the following way

$$\frac{\rho(\mathbf{T}^n)\mathbf{u}^{**} - \rho(\mathbf{T}^n)\mathbf{u}^n}{\Delta t} = \mathbf{F} - \mathbf{M}(\mathbf{u}^*,\mathbf{T}^n)\mathbf{u}^* - \mathbf{B^p}\mathbf{p}^*, \tag{12}$$

where $\rho\mathbf{u}^{**}$ should satisfy the continuity equation. The new value of the pressure $\mathbf{p}^*$ is not known yet, but we can get around this problem by taking the divergence of (12), and applying the continuity equation to the new velocity field $\mathbf{u}^{**}$. The equation for the new pressure will look like

$$-\mathbf{B^u B^p p}^* = -\mathbf{B^u}\left[\mathbf{F} - \mathbf{M}(\mathbf{u}^*, \mathbf{T}^n)\mathbf{u}^* + \frac{\rho(\mathbf{T}^n)\mathbf{u}^n}{\Delta t}\right]. \tag{13}$$

The matrix $\mathbf{B^u B^p}$ is a Laplace-type operator with Neumann boundary conditions. We shall examine it further in the next subsection. After the new pressure has been found we can apply (12) to correct the velocity field. At the next step we compute the new temperature

$$\frac{C_p\rho(\mathbf{T}^n)\mathbf{T}^{n+1} - C_p\rho(\mathbf{T}^n)\mathbf{T}^n}{\Delta t} = -\mathbf{E}(\mathbf{u}^{**}, \mathbf{T}^n)\mathbf{T}^{n+1}. \tag{14}$$

In his paper Issa showed that two correction steps give the optimal new value of the velocity for the time-dependent computations. Although we are dealing with steady-state computations, numerical tests show that two correction steps are also more efficient in the steady-state case. The second correction step will then look like

$$\frac{\rho(\mathbf{T}^{n+1})\mathbf{u}^{n+1} - \rho(\mathbf{T}^n)\mathbf{u}^n}{\Delta t} = \mathbf{F} - \mathbf{M}(\mathbf{u}^{**}, \mathbf{T}^{n+1})\mathbf{u}^{**} - \mathbf{B^p p}^{n+1},$$

and the corresponding pressure system is

$$-\mathbf{B^u B^p p}^{n+1} = -\mathbf{B^u}\left[\mathbf{F} - \mathbf{M}(\mathbf{u}^{**}, \mathbf{T}^n)\mathbf{u}^{**} + \frac{\rho(\mathbf{T}^n)\mathbf{u}^n}{\Delta t}\right].$$

The PISO method can be now formulated as follows:

0. Let $\mathbf{u}^n$, $\mathbf{T}^n$, $\mathbf{p}^n$ be given
1. If the residuals

$$\mathbf{r^u} := \mathbf{F} - \mathbf{M}(\mathbf{u}^n, \mathbf{T}^n)\mathbf{u}^n - \mathbf{B^p p}^n,$$
$$\mathbf{r^T} := -\mathbf{E}(\mathbf{u}^n, \mathbf{T}^n)\mathbf{T}^n$$

are not small enough proceed to step 2.
2. Compute $\mathbf{u}^*$ from

$$\frac{\rho(\mathbf{T}^n)\mathbf{u}^* - \rho(\mathbf{T}^n)\mathbf{u}^n}{\Delta t} = \mathbf{F} - \mathbf{M}(\mathbf{u}^n, \mathbf{T}^n)\mathbf{u}^* - \mathbf{B^p p}^n$$

3. Solve

$$-\mathbf{B^u}\mathbf{B^p}\mathbf{p}^* = -\mathbf{B^u}\left[\mathbf{F} - \mathbf{M}(\mathbf{u}^*, \mathbf{T}^n)\mathbf{u}^* + \frac{\rho(\mathbf{T}^n)\mathbf{u}^n}{\Delta t}\right]$$

4. Compute $\mathbf{u}^{**}$ from

$$\frac{\rho(\mathbf{T}^n)\mathbf{u}^{**} - \rho(\mathbf{T}^n)\mathbf{u}^n}{\Delta t} = \mathbf{F} - \mathbf{M}(\mathbf{u}^*, \mathbf{T}^n)\mathbf{u}^* - \mathbf{B^p}\mathbf{p}^*$$

5. Compute $\mathbf{T}^{n+1}$ from

$$\frac{C_p\rho(\mathbf{T}^n)\mathbf{T}^{n+1} - C_p\rho(\mathbf{T}^n)\mathbf{T}^n}{\Delta t} = -\mathbf{E}(\mathbf{u}^{**}, \mathbf{T}^n)\mathbf{T}^{n+1}$$

6. Solve

$$-\mathbf{B^u}\mathbf{B^p}\mathbf{p}^{n+1} = -\mathbf{B^u}\left[\mathbf{F} - \mathbf{M}(\mathbf{u}^{**}, \mathbf{T}^n)\mathbf{u}^{**} + \frac{\rho(\mathbf{T}^n)\mathbf{u}^n}{\Delta t}\right]$$

7. Compute $\mathbf{u}^{n+1}$ from

$$\frac{\rho(\mathbf{T}^{n+1})\mathbf{u}^{n+1} - \rho(\mathbf{T}^n)\mathbf{u}^n}{\Delta t} = \mathbf{F} - \mathbf{M}(\mathbf{u}^{**}, \mathbf{T}^{n+1})\mathbf{u}^{**} - \mathbf{B^p}\mathbf{p}^{n+1}$$

8. Increase n; return to step 1

4.2 The system for the pressure

In the previous subsection we derived the system for the pressure, the matrix of this system being a product of two matrices $\mathbf{B^u}$ - discrete divergence and $\mathbf{B^p}$ - discrete gradient. Let us determine how this matrix acts on a pressure vector in the two-dimensional case. Define

$$\mathbf{B^p} := \begin{pmatrix} \mathbf{B}_x^{\mathbf{p}} \\ \mathbf{B}_y^{\mathbf{p}} \end{pmatrix},$$

where the block matrices $\mathbf{B}_x^{\mathbf{p}}$, $\mathbf{B}_y^{\mathbf{p}}$ are the approximations of the components of the gradient operator. Bearing this in mind we have

$$\begin{aligned}(-\mathbf{B^u}\mathbf{B^p}p)_C &= h_y[(\mathbf{B}_x^{\mathbf{p}}\mathbf{p})_W - (\mathbf{B}_x^{\mathbf{p}}\mathbf{p})_C] + h_x[(\mathbf{B}_y^{\mathbf{p}}\mathbf{p})_S - (\mathbf{B}_y^{\mathbf{p}}\mathbf{p})_C] = \\ &\frac{h_y}{h_x}[(p_C - p_W) + (p_C - p_E)] + \frac{h_x}{h_y}[(p_C - p_S) + (p_C - p_N)] = \\ &2\left(\frac{h_y}{h_x} + \frac{h_x}{h_y}\right)p_C - \frac{h_y}{h_x}p_W - \frac{h_y}{h_x}p_E - \frac{h_x}{h_y}p_S - \frac{h_x}{h_y}p_N\end{aligned}$$

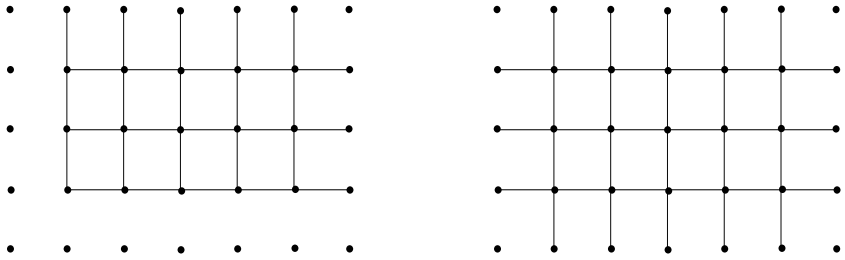

Figure 6: Connectivity graph of the pressure matrix (left) and discrete Laplacian (right)

This is a standard 5-point finite volume (or which is the same in this case, finite element) discretisation of the Laplace operator. It can be shown that the discrete pressure system differs from the 5-point approximation of the Laplace operator only at the boundary nodes. Connectivity graphs of the pressure matrix and the discrete Laplacian are shown in Figure 6.

4.3 Compatibility condition for the pressure system

By construction, the pressure matrix is symmetric, positive-semidefinite, since the diagonal entries are positive, the off-diagonal entries are negative and the row sums are equal to zero. Furthermore this matrix is singular, and the kernel is formed by the constant vectors. Thus before solving the system we need to check whether it is compatible. The compatibility condition can be derived by summing up all equations of the pressure system

$$\sum_C (-\mathbf{B^u B^p})\mathbf{p})_C = \sum_C (-\mathbf{B^u}[\mathbf{F} - \mathbf{M}(\mathbf{u},\mathbf{T})\mathbf{u} + \frac{\rho(\mathbf{T})\mathbf{u}}{\Delta t}])_C. \qquad (15)$$

Since the matrix is symmetric and all row sums are zero, the left-hand side of the equality is zero. At the boundary point C we have

$$\left(\mathbf{F} - \mathbf{M}(\mathbf{u},\mathbf{T})\mathbf{u} + \frac{\rho(\mathbf{T})\mathbf{u}}{\Delta t}\right)_C = \left(\frac{\rho(\mathbf{T})\mathbf{u}}{\Delta t}\right)_C,$$

whence

$$\sum_C (\frac{\rho(\mathbf{T})\mathbf{u}}{\Delta t})_C =$$

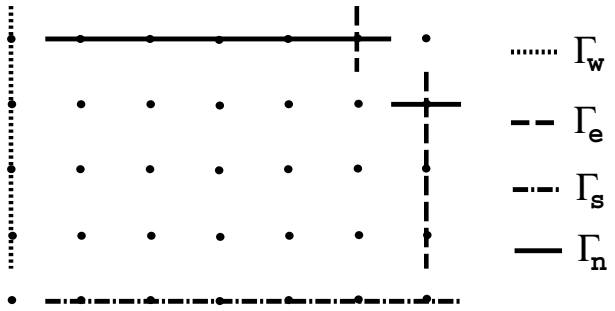

Figure 7: Boundary points involved in the compatibility condition

$$\frac{1}{\Delta t}[\sum_{\Gamma_w} h_y(\rho(T)u)_C - \sum_{\Gamma_e} h_y(\rho(T)u)_C + \sum_{\Gamma_s} h_x(\rho(T)v)_C - \sum_{\Gamma_n} h_x(\rho(T)v)_C].$$

This expression should be equal to zero. After multiplying both sides byΔt we obtain the compatibility condition

$$\sum_{\Gamma_w} h_y(\rho(T)u)_C - \sum_{\Gamma_e} h_y(\rho(T)u)_C + \sum_{\Gamma_s} h_x(\rho(T)v)_C - \sum_{\Gamma_n} h_x(\rho(T)v)_C = 0. \quad (16)$$

Summation is done over four sets of the boundary points, as is shown in Figure 7. The compatibility condition can be interpeted as an approximation of

$$\oint_{\partial\Omega} (\rho(T)\mathbf{u}, \mathbf{n}) d\Gamma = 0,$$

which in turn means that no mass is generated inside the domain. A consequence of (16) is the following. Let us assume that the exact solution is known and satisfies the continuity equation. We use it to obtain the boundary values; but since they also should satisfy (16) it might happen that we cannot use the values of the exact solution at the boundary points. To put it another way, even though the continuous solution satisfies the continuity equation, a test problem constructed from it might not have a solution. The same is true for any solution method for the Stokes/Navier-Stokes equations which makes use of the pressure matrix, though the compatibility condition will perhaps look less "exotic" than for the method considered. One of the reasons why (16) (or its variants) is sometimes ignored, is the correction of the pressure system. The pressure system is singular and since the pressure is determined up to a constant, we can fix the pressure at one point, which

means that the solution is fully determined despite the fact that the original system might be incompatible. The result of that will be the new velocity field that does not satisfy the continuity equation. The last remark about the compatibility condition is that it is not known a priori, but follows from the discrete momentum and the continuity equations.

5 Local defect correction

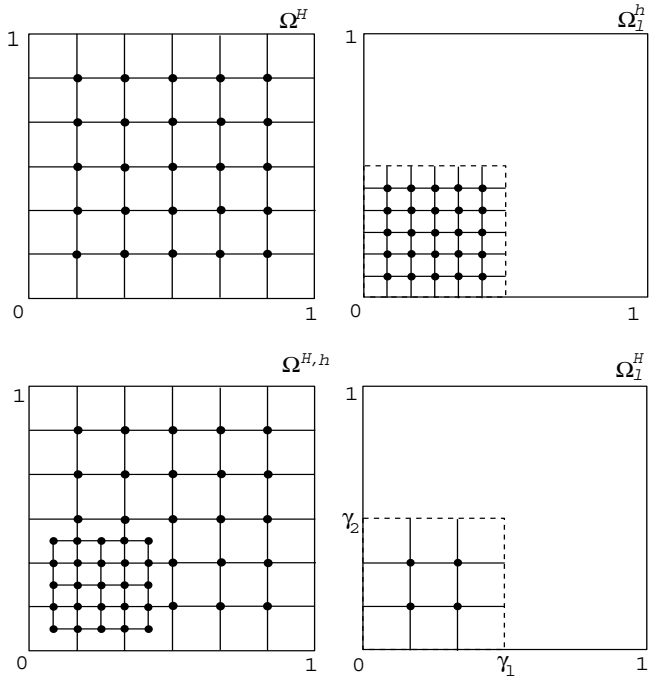

Figure 8: The uniform grids Ω^H, Ω_l^h and Ω_l^H, and the composite grids $\Omega^{H,h}$ for $H = 1/6$, $\sigma = 2$ and $\gamma_1 = \gamma_2 = 1/2$.

Due to the local nature of some processes in the oven, we need a higher resolution in areas where such processes take place. The method we use for local refinement is called *Local Defect Correction* (LDC) [2]. LDC is an iterative procedure which is able to accurately combine solutions on finer local and coarser global grids. In order to describe the method we consider

a model boundary value problem

$$Lu = f \text{ in } \Omega = (0,1) \times (0,1), \tag{17}$$
$$u = \varphi \text{ on } \partial\Omega. \tag{18}$$

Here L is an arbitrary elliptic operator, f is a given function on Ω, φ a given function on the boundary $\partial\Omega$.

The model composite grid $\Omega^{H,h}$ (see Fig.8)is composed of a global coarse grid and a local fine grid. The *global coarse grid* Ω^H is a uniform grid with grid size H

$$\Omega^H = \{(x_i, y_j) |\ x_i = iH,\ \ y_j = jH,\ \ 0 < i < 1/H,\ \ 0 < j < 1/H\}.$$

We suppose that $1/H \in \mathbf{N}$. The *local fine grid* Ω^h_{loc} is a uniform grid with grid size $h < H$, covering the region of local refinement $\Omega_{loc} = (0, \gamma_1) \times (0, \gamma_2) \subset \Omega$

$$\Omega^h_{loc} = \{(x_i, y_j) |\ x_i = ih,\ \ y_j = jh,\ \ 0 < i < 1/h,\ \ 0 < j < 1/h\}.$$

We assume $\gamma_1/H \in \mathbf{N}$, $\gamma_2/H \in \mathbf{N}$ and $H/h \in \mathbf{N}$. The interface Γ is defined as the part of the boundary $\partial\Omega_{loc}$ that lies inside Ω,

$$\Gamma = \partial\Omega_{loc} \cap \Omega.$$

We assume also that the high activity region of the boundary value problem lies inside the subregion Ω_{loc}. The *composite grid* $\Omega^{H,h}$ is defined by

$$\Omega^{H,h} = \Omega^H \cup \Omega^h_{loc}.$$

The refinement factor σ is defined as the ratio of the coarse grid size H and the fine grid size h,

$$\sigma = H/h.$$

Besides the uniform coarse grid Ω^H and the uniform local grid Ω^h_{loc}, there is one more important grid, namely the *local coarse grid* Ω^H_{loc} which is defined by

$$\Omega^H_{loc} := \Omega^H \cap \Omega^h_{loc}$$

In the LDC method one starts by discretising the boundary value problem on the global coarse grid Ω^H. This yields the basic discretisation

$$L^H u^H_0 = f^H \text{ on } \Omega^H. \tag{19}$$

The grid function u_0^H is an approximation of the solution of the boundary value problem (17), (18). This grid function is used for discretising the boundary value problem on the local fine grid Ω_{loc}^H. Boundary conditions for this problem are obtained from u_0^H by means of interpolation. The discrete problem on the local fine grid is denoted by

$$L_{loc}^h u_{loc,0}^h = f_{loc}^h(u_0^H) \text{ on } \Omega_{loc}^h. \tag{20}$$

The dependence of f_{loc}^h on the aproximation u_0^H is denoted explicitly in (20). The approximations u_0^H and $u_{loc,0}^h$ are used to define a *composite grid approximation* $u_0^{H,h}$

$$u_0^{H,h}(x) = \begin{cases} u_{loc,0}^h(x) & x \in \Omega_{loc}^h \\ u_0^H(x) & x \in \Omega^{H,h} \backslash \Omega_{loc}^h \end{cases}.$$

In the local defect correction method the fine grid approximation $u_{loc,0}^h$ is used to correct the basic discretisation (19) in the following way. The global coarse grid approximation u_0^H and the local fine approximation $u_{loc,0}^h$ are combined to define the global coarse grid function w^H,

$$w^H(x) = \begin{cases} u_{loc,0}^h(x) & x \in \Omega_{loc}^H \\ u_0^H(x) & x \in \Omega^H \backslash \Omega_{loc}^h \end{cases}.$$

Substituting this function into the basic discretisation yields a *residual grid function* or *defect*,

$$d^H = L^H w^H - f^H.$$

The values of this defect at grid points inside Ω_{loc} are used to update the right hand side f^H of the basic discretisation,

$$\bar{f}^H(x) := \begin{cases} f^H(x) + d^H(x) & x \in \Omega_{loc}^H \\ f^H(x) & x \in \Omega^H \backslash \Omega_{loc}^h \end{cases}.$$

The updated coarse grid problem reads

$$L^H u_1^H = \bar{f}^H(u_{loc,0}^h, u_0^H) \text{ on } \Omega^H. \tag{21}$$

Equation (21) yields an approximation u_1^H of the solution of the boundary value problem (17), (18) on the global coarse grid. Like the approximation u_0^H, the approximation u_1^H is used to define artificial Dirichlet boundary values on the interface. The related discrete problem on the local fine grid reads

$$L_{loc}^h u_{loc,1}^h = f_{loc}^h(u_1^H) \text{ on } \Omega_{loc}^h.$$

The approximations u_1^H and $u_{l,1}^h$ are used to define a composite grid approximation $u_1^{H,h}$ of the solution of the boundary value problem

$$u_1^{H,h}(x) = \begin{cases} u_{loc,1}^h(x) & x \in \Omega_{loc}^h \\ u_1^H(x) & x \in \Omega^{H,h} \backslash \Omega_{loc}^h \end{cases} .$$

The LDC method can be summarised as the following scheme:

1. Solve

$$L^H u_n^H = f_n^H \text{ on } \Omega^H, \;\; f_0^H = f^H.$$

2. Solve

$$L_{loc}^h u_{loc,n}^h = f_{loc,n}^h(u_n^H) \text{ on } \Omega_{loc}^h.$$

(Boundary conditions for this problem are obtained from u_n^H by means of interpolation.)

3. Construct the composite approximation $u_n^{H,h}(x)$

$$u_n^{H,h}(x) = \begin{cases} u_{loc,n}^h(x) & x \in \Omega_{loc}^h, \\ u_n^H(x) & x \in \Omega^{H,h} \backslash \Omega_{loc}^h \end{cases} .$$

4. Construct the global coarse grid function w^H

$$w^H(x) = \begin{cases} u_{loc,n}^h(x) & x \in \Omega_{loc}^H, \\ u_n^H(x) & x \in \Omega^H \backslash \Omega_{loc}^h \end{cases} .$$

5. Compute the defect d^H

$$d^H = L^H w^H - f^H.$$

6. Update the right-hand part of the global problem

$$f_{n+1}^H(x) = \begin{cases} f_n^H(x) + d^H(x) & x \in \Omega_{loc}^H, \\ f_n^H(x) & x \in \Omega^H \backslash \Omega_{loc}^h \end{cases} .$$

7. Check convergence; return to step 1, if not accurate enough.

6 Stirred flow

One of the situations where we can use the advantages of LDC is the modelling of a stirrer. Usually the stirrer is located in a so-called *doghouse*, a

small tank from which glass flows to production lines. We model the stirrer as a small rectangular solid piece, with velocity vectors prescribed in such a way that they move the flow around. The problem with computations involving a stirrer is that the size of the stirrer is normally much smaller than the characteristic length. The standard way to resolve the problem is to refine globally in the vicinity of the stirrer. Results of this approach are shown in Figure 9 (since original computations are carried out in 3D, we plot the results for a fixed vertical coordinate) This approach gives the best possible

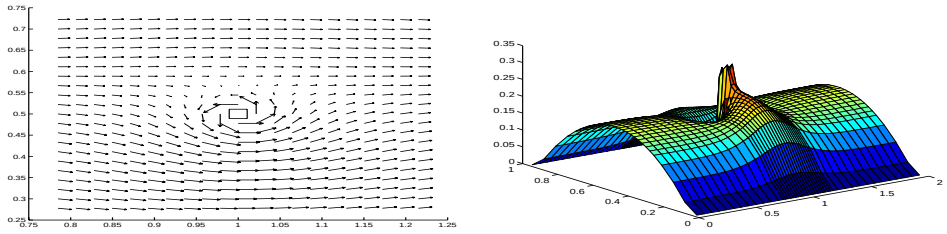

Figure 9: Solution obtained by means of non-uniform global refinement - magnified velocity field (left), absolute value of velocity (right) (horizontal crossection)

accuracy, but has some strong disadvantages. First, by refining globally we obtain a considerable amount of grid points, in which much the information is irrelevant. In 3D this leads to a significant and superfluous increase in memory and computational time. Secondly, the condition number of the discrete system depends on the mesh size ratio.

We can avoid this problems by refining locally, and use LDC. The result is depicted in Figure 10. One of the main advantages of LDC is that we don't need a special data structure for the composite grid, since it is constructed of regular global and local grids. Furthermore, since the systems used in LDC are of smaller size and better conditioned due to smaller mesh size ratios, the solution process of linear systems is considerably more efficient. Though LDC is an iterative procedure, it converges very fast. We use the results from the global refinement $\hat{\mathbf{u}}$ in order to estimate convergence and accuracy of LDC. $\mathbf{u}^{H,h}$ denotes the solution on the composite grid (see Table 1).

Another specific feature of LDC is the use of the defect. Via the defect local pertubations affect the solution globally. If we would stop the procedure after the initial step, that is without updating the global problem, the results

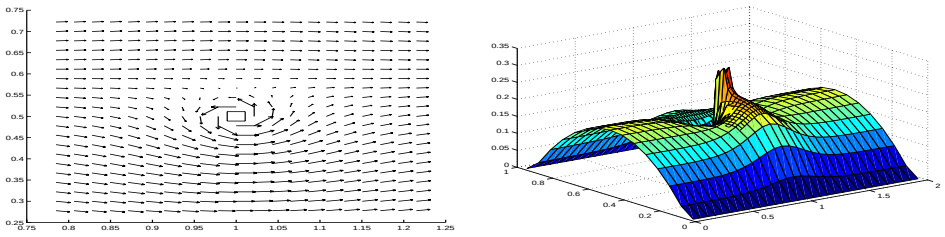

Figure 10: Solution obtained with LDC - magnified velocity field (left), absolute value of velocity (right) (horizontal cross section)

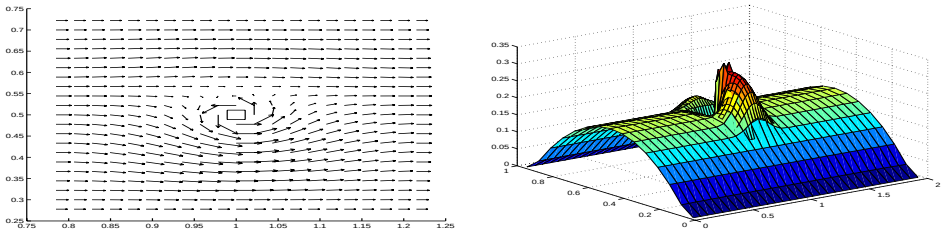

Figure 11: Solution after zero iteration - magnified velocity field (left), absolute value of velocity (right) (horizontal cross section)

would not be accurate enough, see Figure 11

7 Bubbling

Another situation in the glass oven where we can successfully use LDC, is a bubbling process. The air bubbles are injected into the glass tank from the bottom, and while moving to the top, they attract other bubbles, thus removing the air from the glass.

Suppose we have with an initial volume per bubble V_0. Since the hydrostatic pressure decreases going from the bottom to the top, the bubble diameter changes to

$$V_b = V_0 \frac{1 + \rho g H/p_0}{1 + \rho g(H - z)/p_0},$$

where H is a total glass height, z the glass height and p_0 the athmospheric pressure. Suppose the total bubbling volume flow is Q. Then the distance

#LDC iter.	$\frac{\|\hat{\mathbf{u}}-\mathbf{u}^{H,h}\|_\infty}{\|\hat{\mathbf{u}}\|_\infty}$
0	$7.90 \cdot 10^{-2}$
1	$3.79 \cdot 10^{-2}$
2	$2.15 \cdot 10^{-2}$
3	$1.26 \cdot 10^{-2}$

Table 1: LDC iteration results

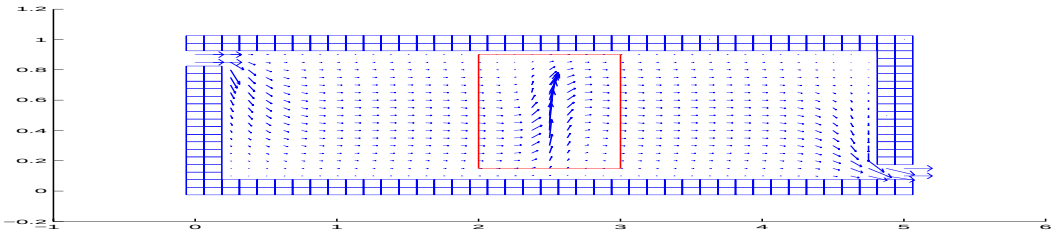

Figure 12: Global velocity field (without LDC) (vertical cross section)

between the bubbles is $d = vV_0/Q$, where v is the bubble rising velocity, i.e. the sum of the undisturbed vertical glass velocity and the relative glass-bubble velocity. From one bubble, a force acting on the glass is

$$F_b = -V_b \rho g.$$

Since per vertical meter, there are on the average 1/d bubbles pushing on the glass, the average force per meter height is:

$$F_a = F_b/d = -\frac{V_b \rho g}{vV_0} Q.$$

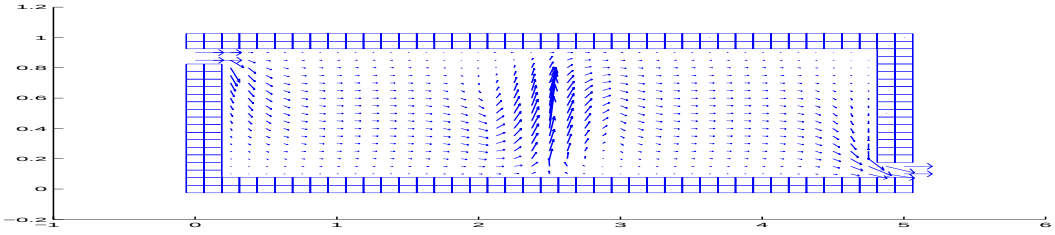

Figure 13: Global velocity field (with LDC) (vertical cross section)

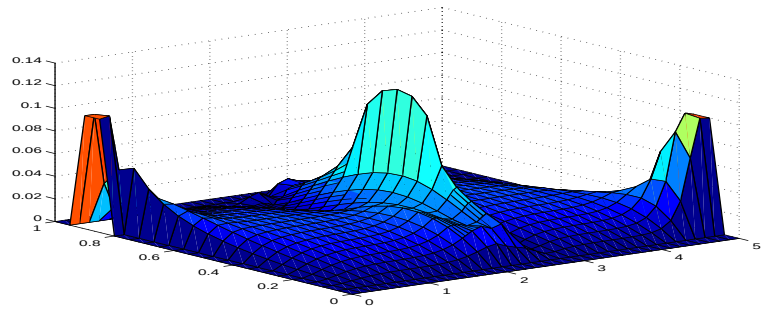

Figure 14: Absolute value of the velocity (vertical cross section)

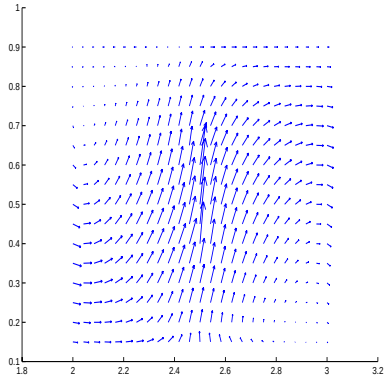

Figure 15: Local velocity field (vertical cross section)

After that we correct the right-hand side of the vertical momentum equation as follows:

$$F_C = h_z F_a - h_x h_y h_z \rho(T_c) g$$

The computations on a coarse grid appear to be not accurate enough, see Fig.12. In order to improve the solution we use LDC, see Fig.13. The absolute value of the composite solution and the velocity field in the region of the local refinement are depicted in Figure 14 and 15 respectively.

References

[1] G.K.Batchelor, *An introduction to fluid dynamics*, Cambridge University Press, (1970).

[2] P.J.J.Ferket and A.A.Reusken, *Further analysis of the local defect correction method*, Computing , 56(1996), pp.117-139.

[3] J.H.Ferziger, M.Peric, *Computational methods for fluid dynamics*, Springer (1996).

[4] R.I.Issa, *Solution of the Implicitly Discretised Fluid Flow Equations by Operator-Splitting*, Journal of computational physics, 62 (1985), pp.40-65.

[5] P.K.Kundu, *Fluid Mechanics*, Academic Press, San-Diego (1990).

[6] S.Nefedov, *Simulation of glass flow in an oven*, in Progress in Industrial Mathematics at ECMI98, B.G.Teubner, (1999), pp.106-113.

[7] S.V.Patankar, *Numerical heat transfer and fluid flow*, Hemispere Publishing Corporation (1980).

[8] A.Quarteroni and A.Valli, *Numerical approximation of partial differential equations*, Springer (1994).