Workshop on Virtual Reality Interaction and Physical Simulation VRIPHYS (2009)
H. Prautzsch, A. Schmitt, J. Bender, M. Teschner (Editors)

Smoke Simulation for Fire Engineering using a Multigrid Method on Graphics Hardware

S. L. Glimberg[1] and K. Erleben[1] and J. C. Bennetsen[2]

[1]eScience Center, Department of Computer Science, University of Copenhagen, Denmark
[2]Rambøll Denmark A/S

Abstract
We present a GPU–based Computational Fluid Dynamics solver for the purpose of fire engineering. We apply a multigrid method to the Jacobi solver when solving the Poisson pressure equation, supporting internal boundaries. Boundaries are handled on the coarse levels, ensuring that boundaries will never vanish after restriction. We demonstrate cases where the multigrid solver computes results up to three times more accurate than the standard Jacobi method within the same time. Providing rich visual details and flows closer to widely accepted standards in fire engineering. Making accurate interactive physical simulation for engineering purposes, has the benefit of reducing production turn-around time. We have measured speed-up improvements by a factor of up to 350, compared to existing CPU-based solvers. The present CUDA-based solver promises huge potential in economical benefits, as well as constructions of safer and more complex buildings. In this paper, the multigrid method is applied to fire engineering. However, this is not a limitation, since improvements are possible for other fields as well. Traditional Jacobi solvers are particulary suitable for the methods presented.

Categories and Subject Descriptors (according to ACM CCS): Computer Graphics [I.3.1]: Graphics processors—Computer Graphics [I.3.1]: Parallel processing—Computer Graphics [I.3.5]: Physically based modeling—Computer Graphics [I.3.7]: Animation—Mathematics of Computing [G.1.8]: Multigrid and multilevel methods—

Keywords: Smoke simulation, Fire engineering, GPU, Multigrid solver

1. GPU Acceleration for Fire Engineering

The implementation and use of performance-oriented applications and regulations within recent years, has lead to an increased design freedom for constructing more spectacular and complex buildings. These buildings push the structural design to the limit of what is possible, but also the structural response during a fire and the needs for improving the safety and reliability of buildings in such an event. In general, the main purpose of Computational Fluid Dynamics (CFD) simulations in this context, is to predict the spread of smoke through a building space and the effectiveness of smoke ventilation to restrict that spread. The results are then assessed using some form of "tenability" criteria, typically based on a combination of visibility and temperature. All of this is used as a means of estimating the "available safe egress time".

Our approach to implement a fast solver for smoke propagation is therefore very attractive. There has been several approaches to construct fast smoke solvers for computer graphics [Sta99, Har04, KC07], but only a very few for actual smoke propagation for engineering purposes. However, interest in fast CFD solvers has increased over the last few years as the available software and hardware have evolved. Our solution can be applied for fast and rough check and thereby evaluate new designs quickly. Furthermore, the speed of the present solver makes it possible to analyse different locations of the fire sources, thereby improving and making the buildings safer within short time. Supporting computational grid sizes of $256 \times 256 \times 256$, makes the current solver very attractable. Within fire engineering, the size of the computational grids range from 500.000 to 4.000.000 cells in most case. If more computational power was available, additional details and larger models could be analyzed. The main limitation is the overall computational time required to complete a simulation, rang-

DOI: 10.2312/PE/vriphys/vriphys09/011-020

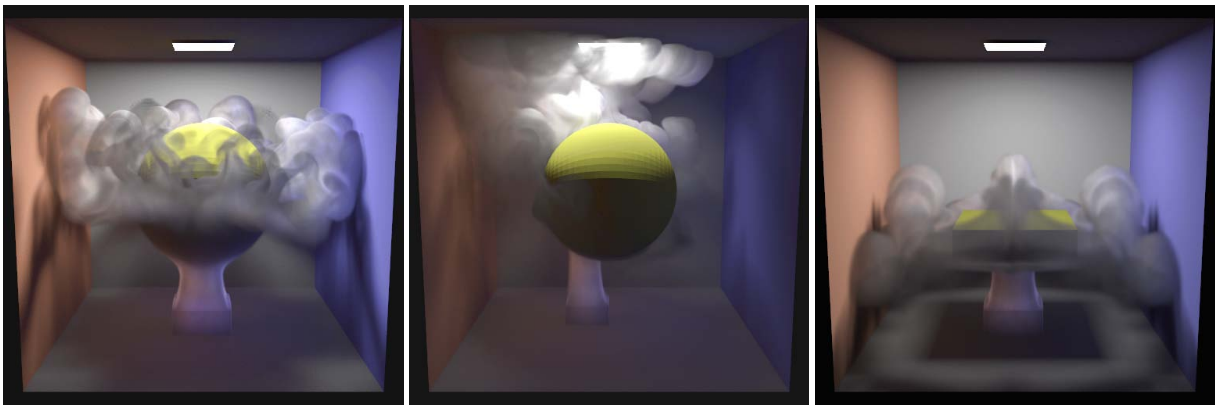

Figure 1: *We have improved state-of-art computer graphics methods for computational fluid dynamics on GPUs with a multigrid method. The improved interactive physical simulation method yields an accuracy improvement that is up to three times better than the standard Jacobi method, with the same time consumption. The improved accuracy makes it possible to do smoke simulation for fire engineering with a speedup factor of 350 on the production turn-around time. The images are post-processed using photon mapping and smoke data from three various setups.*

ing from several hours to several days, even when a parallel simulation is applied on a cluster of computers.

Fire engineering is in use every day in engineering businesses, and the application of computer simulation is necessary in order to provide information about the safety level inside a building. Software for fire engineering simulation, regarding smoke propagation, includes the Fire Dynamics Simulator (FDS) developed by NIST [fds09]. Due to the industrial application of our work [Gli09], the main focus is on a fast, interactive smoke propagation. Both the present solver and FDS apply uniform regular grids. Thus, we disregard smoothed particle hydrodynamics methods [BT07], methods for unstructured grids [KFCO06, ETK*08], and lattice Boltzmann methods [LFWK06].

Many works have been done in computer graphics using computational fluid dynamics on regular grids for animation [BMF07, Bri08]. An Eulerian approach to 3D fluid simulation was used in [FM96, FM97] demonstrating advantages over earlier work using particle systems, and 2D simulations. In [Sta99] a semi-Lagrangian implicit time stepping method was introduced to deal with convection. In [FF01] a liquid surface was tracked using a combination of particles and a signed distance field. A physically consistent vorticity confinement term to model small scale rolling features of smoke were used by [FSJ01]. Large scale simulation were addressed in [RNGF03] using high resolution tiled 2D velocity fields. To capture the small scale visual details, octrees were presented in [LGF04]. Numerical methods for handling discontinuities in density and viscosity field, as well as pressure jump conditions, were developed in [HK05]. A projection method was introduced by [LSSF06] to deal with multiple interacting fluid flows. To counter the excessive false diffusion made by the stable fluid method, an unconditionally stable MacCormack method was presented in [SFK*08].

Most previous work focuses on producing animations for film production with rich visual detail [MCPN08] or realistic game scenes [KC07]. They prefer visual quality on the expense of correct physical behavior of the flow. The work [Sta99] inspired a long range of GPU based fluid solvers, exploiting the massive parallelism of the modern graphics hardware architecture. The interactive frame rates, obtained with these computer graphics solutions, imply that one can build pre-production tools for reducing turn-around time in other application areas than graphics. Thus, we have chosen to concentrate on GPU acceleration using CUDA, in order to create a feasible production tool for fire engineering. A wide variety of test cases exist for CFD verification. In this paper we have used the results presented by Ghia et al. [GGS82] as reference.

Multigrid CFD methods in engineering applications are well-known [GGS82, FP02]. However, they do not deal with the issues of making a GPU implementation and handling internal boundaries, which we address in this work. GPU-based multigrid methods for CFD is slowly starting to appear. To our knowledge, the only other work similar to ours, is by Molemaker et. al. [MCPN08, MC09]. They present a staggered approach, using a QUICK advection scheme in combination with an algorithm called Iterated Orthogonal Projection. Solid boundaries are ignored at the course levels of the multigrid solver. Though the authors guarantee convergence, ignoring boundaries may lead to spurious behavior near thin obstacles. In fact, we demonstrate in Figure 2 that ignoring boundaries in the pressure field may lead to very incorrect flow behavior.

2. The GPU Friendly Method

Fast and rough check of smoke propagation for fire engineering is of great interest [Gli09]. Thus, we do not consider compressibility, smoke particles, combustion, nor turbulence models etc. Instead it suffices to solve the incompressible version of the Navier-Stokes as partial differential equations. Treating smoke as an incompressible fluid, is a fair assumption when it moves at velocities apparent at fire scenes. The Navier-Stokes equations as solved in this paper have the form,

$$\frac{\partial u}{\partial t} = -(u \cdot \nabla) u + \nu \nabla^2 u - \frac{1}{\rho}\nabla p + f, \tag{1a}$$

$$\nabla \cdot u = 0, \tag{1b}$$

where u is the velocity field, ν is the dynamic viscosity, ρ is the constant density, p is the pressure field, f is the external forces, and ∇ is the spatial differential operator.

The primary computational effort in most CFD solvers is solving linear systems for diffusion and pressure correction as outlined below. Standard computer graphics GPU-based methods use an iterative Jacobi method to solve the systems of linear equations [Har04, KC07], due to ease of implementation on a GPU. However, as we will explain below and show later in Section 3.1, the poor convergence rate of the Jacobi method makes it unattractive for interactive purposes. The Gauss-Seidel method is known to have better convergence properties than the Jacobi method. However, the convergence rate is linear [Saa03] as for the Jacobi method, so little is gained. Besides, parallelized operations call for a red-black Gauss-Seidel method, which is more difficult to implement than the simple Jacobi method. Methods such as Conjugate Gradient converges faster than the Jacobi method [Saa03], but are not as easily implemented on the GPU. The multigrid method has the advantage that it converges well, and that any iterative solver can be reused, minimizing the implementation modifications. Thus, in our work we have utilized a multigrid approach. In the following we will first give a short presentation of the fractional step approach for solving the Navier-Stokes equations, then we outline the standard Jacobi method, which serves as the basis for our multigrid extension presented in Section 2.1.

Higher order schemes or staggered grids are less beneficial, compared to more accurate solutions to the Poisson equation. Adaptive grids are disregarded as they are not easily mapped to the GPU. A collocated regular grid representation of all fields has been chosen, because it fits the GPU programming model well. Spatial derivatives are approximated using first order central differences. The discrete gradient at location i, j, k of a field p is approximated by,

$$\nabla p_{i,j,k} = \begin{bmatrix} \frac{p_{i+1,j,k} - p_{i-1,j,k}}{2\Delta x} \\ \frac{p_{i,j+1,k} - p_{i,j-1,k}}{2\Delta y} \\ \frac{p_{i,j,k+1} - p_{i,j,k-1}}{2\Delta z} \end{bmatrix}, \tag{2}$$

where Δx, Δy, and Δz are the cell sizes in their respective direction. Using a fractional step method, each of the four fractions on the right hand side of (1a) is handled individually, yielding four sequential updates at each time step,

$$\frac{\partial u_1}{\partial t} = f, \tag{3a}$$

$$\frac{\partial u_2}{\partial t} = -(u_1 \cdot \nabla) u_1, \tag{3b}$$

$$\frac{\partial u_3}{\partial t} = \nu \nabla^2 u_2, \tag{3c}$$

$$\frac{\partial u_4}{\partial t} = -\frac{1}{\rho}\nabla p. \tag{3d}$$

The continuity constraint from (1b), must be satisfied after the last fractional step, ie. $\nabla \cdot u_4 = 0$. The continuity constraint is upheld using pressure projection presented later. External forces are modeled as the sum of gravity and thermal buoyancy, using the Boussinesq approximation,

$$f = \alpha\rho z + \beta (T - T_0) z, \tag{4}$$

where ρ is the smoke fraction, $z = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$, T_0 is the ambient temperature, and α and β are constant coefficients controlling the contribution of gravity and buoyancy. High temperatures and smoke fractions originate from static smoke sources and are also updated every time step as,

$$\frac{\partial T}{\partial t} = -(u \cdot \nabla) T + \kappa \nabla^2 T, \tag{5a}$$

$$\frac{\partial \rho}{\partial t} = -(u \cdot \nabla) \rho, \tag{5b}$$

where κ is thermal diffusivity. Advection of both the velocity, temperature and smoke fraction fields is applied using the unconditionally stable semi–Lagrangian approach [Sta99]. When solving the diffusion fraction we also use an implicit approach to ensure stability,

$$\left(I - \Delta t \nu \nabla^2\right) u_3 = u_2, \tag{6}$$

where I is the identity matrix and Δt is the discrete time step. This is a system of linear equations, which we solve using the iterative Jacobi method. The velocity field obtained after the first three fractional steps may be divergent. For incompressible flows we use the pressure field to force continuity, ie. create a pressure projection of the divergent velocity field into a divergence–free field. Using explicit Euler on (3d) yields,

$$u_4 = u_3 - \frac{\Delta t}{\rho}\nabla p. \tag{7}$$

The question is how to find a p such that u_4 becomes divergence–free when subtracting the pressure term from u_3. If we apply the divergence operator to both sides of (7), the left hand side will be zero by definition,

$$\underbrace{\nabla \cdot u_4}_{0} = \nabla \cdot u_3 - \frac{\Delta t}{\rho}\nabla^2 p \tag{8a}$$

$$\nabla^2 p = \frac{\rho}{\Delta t}(\nabla \cdot u_3). \tag{8b}$$

The latter equation is a Poisson equation where the only unknown variable is p. Once (8b) is solved for p, we insert it back into (7) and calculate the projection that will make u_4 divergence–free. The two constant density fractions in (7) and (8b) cancel out each other, simplifying the equations. One simulation time step is summarized as,

1. Apply external forces, advection, and diffusion to arrive at u_3.
2. Solve $\nabla^2 p = \nabla \cdot u_3$ for p.
3. Calculate the projection $u_4 = u_3 - \nabla p$ to make u_4 divergence–free.

It should be clear that the solution to the pressure field is related to the satisfaction of the continuity constraint and consequently the presence of dissipation. Hence, there is an obvious motivation to solve the Poisson pressure equation accurately. Using central difference approximations, the computational stencil of a Jacobi iteration becomes,

$$\begin{aligned} p^{n+1}_{i,j,k} = & \frac{p^n_{i+1,j,k} + p^n_{i-1,j,k}}{\alpha(\Delta x)^2} + \frac{p^n_{i,j+1,k} + p^n_{i,j-1,k}}{\alpha(\Delta y)^2} + \\ & + \frac{p^n_{i,j,k+1} + p^n_{i,j,k-1}}{\alpha(\Delta z)^2} - \frac{\nabla \cdot u_3}{\alpha}, \end{aligned} \tag{9}$$

where n represents the iteration number and $\alpha = \frac{2}{(\Delta x)^2} + \frac{2}{(\Delta y)^2} + \frac{2}{(\Delta z)^2}$. Notice that each update only uses the adjacent values in each coordinate direction. Consequently, information is only spread one cell after each update. This reduces convergence when the grid resolution is high and if the true solution is low frequent [BHM00]. These facts motivate to use an alternative approach to reduce inaccurate flow behavior.

2.1. The Multigrid Method

The Jacobi method delivers poor convergence towards the true solution, primarily due to slow information distribution. A poor solution to the pressure field causes the flow to be divergent and thus dissipative. Nevertheless, it has been the primary choice in many applications presented in literature [Har04, KC07]. In this section we present the multigrid technique, using the Jacobi solver as the relaxation scheme. The improvement enables our CFD solver to be more useable for engineering purposes, as results show in Section 3.1. Multigrid has the primary advantage that it fits the GPU programming model well, and it makes any iterative solver reusable. A general description of the multigrid technique can be found in [BHM00].

The basic idea of a multigrid approach is to perform few iterations on restricted systems of linear equations, and then interpolate corrections back into the original system. Remember that in our case, the linear system is given from the pressure and velocity values in the simulation grid. Let Ω_h denote the finest grid, Ω_{h2} the second finest grid, with half as many cells in each direction, and so on. A projection operator $\mathbf{I}$, taking a vector x, from one grid to another is defined as,

$$x_h = \mathbf{I}^h_{h2} x_{h2} \tag{10a}$$

$$x_{h2} = \mathbf{I}^{h2}_h x_h. \tag{10b}$$

For the restriction operation, every second cell in each direction simply remains. Averaging adjacent values are sometimes used, but the cost of reading multiple values from global device memory is not optimal. Thus, we prefer the solution that minimizes access to global memory. Interpolation is a bit more difficult, because most of the values in the fine grid have no equivalents in the coarse grid. As the name suggests, interpolation should be performed for these grid points. In three dimensions, the following four scenarios occur for interpolation to a finer grid.

1. The new value can be copied directly from the coarse grid
2. The new value must be interpolated from two coarse grid points
3. The new value must be interpolated from four coarse grid points
4. The new value must be interpolated from eight coarse grid points

The Poisson equation (8b) is a system of linear equations. Rewriting it into to the general form of linear systems yields,

$$Ax = b. \tag{11}$$

The error ε, between the true solution and the solution after n iterations is related to the residual as,

$$A\varepsilon = b - Ax^n = r. \tag{12}$$

Restricting the residual enables us to express a new system of linear equations, where the error is the unknown,

$$r_{h2} = \mathbf{I}^{h2}_h r_h \tag{13a}$$

$$A_{h2}\varepsilon_{h2} = r_{h2}, \tag{13b}$$

where A_{h2} still corresponds to the Laplacian operator (∇^2). Thus, the matrix is still implicit and is never actually constructed, which enables us to reuse the same Jacobi solver to perform relaxation of ε_{h2}. When a solution to ε_{h2} is found after a few iterations, the fine grid solution can be corrected with the interpolated error as,

$$\tilde{x}_h = x_h + \mathbf{I}^h_{h2}\varepsilon_{h2}, \tag{14}$$

where $\tilde{x}_h$ is the new adjusted guess. This list summarizes the multigrid method using one restriction:

- Relax $A_h x_h = b_h$ with initial guess x^0_h.
- Compute the residual $r_h = b_h - A_h x_h$
- Restrict the residual $r_{h2} = \mathbf{I}^{h2}_h r_h$
- Relax $A_{h2}\varepsilon_{2h} = r_{h2}$
- Interpolate the error $\varepsilon_h = \mathbf{I}^h_{h2}\varepsilon_{h2}$
- Adjust the fine grid solution $\tilde{x}_h = x_h + \varepsilon_h$

- Relax $A_h x_h = b_h$ with the updated $\tilde{x}_h$ as new start guess.

There is no need to stop after just one restriction, in fact multigrid often gives the best results if it continues until the coarsest grid is reached. The whole process might also be repeated two or more times with a different number of restrictions. One repetition of restrictions and interpolations is called a V-cycle. In Section 3.1 we present some comparisons of different V-cycles.

Static obstacles and outer boundaries are both represented using a grid of the same resolution as the simulation domain. A 2D example is given in Figure 3. Using a collocated grid, causes the boundaries to be implicit, meaning that they are located in between discrete values. We use Dirichlet boundary conditions to control the implicit values. To handle corners and edges we calculate an average, based on the n adjacent non-boundary cells. The pressure term should not cause acceleration across the boundary, so the gradient at the boundaries should be zero. This is achieved with the following Dirichlet condition,

$$p_{\text{in}} = \frac{1}{n}\sum_{i=0}^{n} p^{i}_{\text{out}}, \tag{15}$$

where "in" represents the discrete value inside the boundary cell and "out" represents the values outside the boundary. Upholding boundary conditions should be performed after any alternation of the pressure field, to ensure that pressure is not built up across boundaries. If boundaries are ignored at the restricted levels of the multigrid method, pressure is interpolated back across boundaries, causing the flow to travel through obstacles. Figure 2 illustrates the problem. If the boundary field is naively restricted along with the pressure field, thin obstacles might vanish in the coarse grids, resulting in the same problem. Our solution is to let boundaries dominate the restriction process. In three dimensions, eight cells becomes one after restriction. If any of these eight cells are marked as a boundary, then the restricted cell will also be a boundary cell. The difference between naive and dominant restriction is illustrated in Figure 3.

Dominant restriction of the boundaries ensures that they will never vanish, but it does not ensure that boundaries become one cell thick. This is sometimes a problem in combination with collocated grids, because there is only one cell to represent the two implicit boundaries on each side. We have not handled the issue in this work, and will therefore avoid setups where such cases would appear. In fire engineering, this is rarely an issue because thin obstacles are not common. Besides, higher resolutions remove the issue.

Our CFD solver was implemented using the CUDA programming model, targeting the highly parallel architecture of the GPU. Care must be taken to achieve optimal GPU performance. There are several restrictions and guidelines on how to acces device memory [NVI09]. CUDA threads are organized in block structures, which is again organized in grids. To our knowledge no best practice has been presented

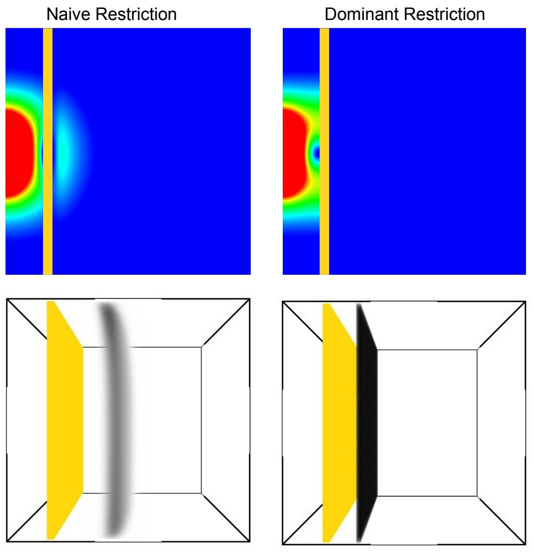

Figure 2: *A ventilator is placed to left while an obstacle splits the room in two. On the left side of the obstacle there is a passive smoke layer. The top images illustrate the slice from the center of the velocity field after two time steps. The bottom images illustrate the smoke propagation after 1 second of simulation. Naive restriction causes the flow to penetrate the obstacle, creating incorrect movement on the right side of the obstacle.*

on how to set up CUDA blocks and grids for 3D problems. We have chosen to allocate all scalar and vector fields using the 3D structures provided by CUDA. Thus, a kernel call should process each entry in the 3D field once. To achieve this, we organize threads in a 1D block with the same size as the x-resolution of the fields. Blocks are then organized in a 2D grid, with dimensions equal to the y- and z-resolutions.

The total amount of threads then add up to the same as the resolution of the simulation grid. This approach is very flexible, it requires minimal index calculations in the kernels, and it allows variables along the x-direction to be written into shared memory for fast access. This is useful when for example finite difference operations are performed. Another approach was presented in [BP08], potentially allowing more variables to be written into shared memory. Unfortunately their approach requires complex index calculations and the acces pattern from global memory does not respect the guide lines given in [NVI09]. Consequently we believe our approach to be superior, though comparisons hereof have not been performed.

Current GPUs have a limitation of 512 threads per block, thus a natural limitation for simulation resolutions arises at this point using our approach. However, another problem of limited device memory also arise around this point, for even

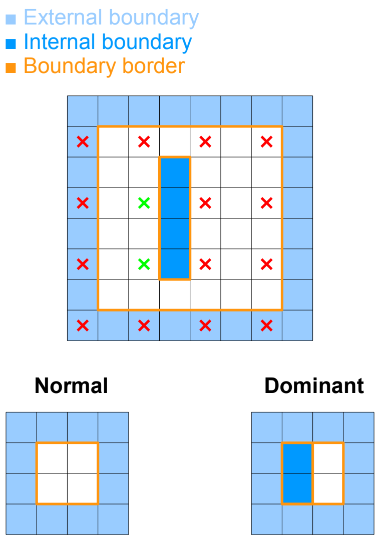

Figure 3: *2D illustration indicating the implicit boundaries and the difference between naive and dominant restriction. The 16 crosses indicate the cells that are restricted to the coarser level. Since none of the crosses are internal boundaries, naive restriction causes the obstacle to vanish. For dominant restriction, the green crosses are considered boundary cells and thus the obstacle remains.*

the most modern GPUs with 4GB of memory. We therefore find this limitation to be acceptable.

3. The Multigrid Method Accuracy Improvements

We first present the results obtained using the multigrid approach. Results are compared to the traditional Jacobi method. For the solver to be useful for engineering purposes, it must handle real-like scenarios within reasonable accuracy. Reference results have been presented in literature for many years, we present a comparison of our results to the results presented by Ghia et al. [GGS82].

3.1. The Multigrid Method vs the Jacobi Method

To compare the multigrid method with the simple Jacobi method, we set up an initial divergent velocity field. The Poisson pressure equation solves for a pressure field that removes this divergence. Hence, good solutions to the pressure field will cause the velocity field to be divergence-free within less iterations. The multigrid approach delivers a set of parameters, such as the number of restrictions, iterations at each level and the number of repeated V-cycles. In this test we use a $128 \times 128 \times 128$ grid resolution and vary the number of restrictions. The maximum number of restrictions used is four, which cause the coarsest grid resolution to be

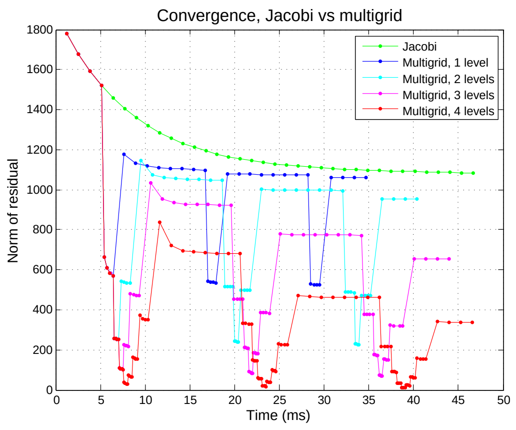

Figure 4: *The norm of the residual, calculated after each iteration for both the Jacobi method and the multigrid method. The peaks in the multigrid plots are caused by the restrictions and interpolations, which change the sizes of the residuals, and thus the norms. Notice how every V-cycle cuts of an considerable amount of the norm when it returns to the finest grid. The most restrictions yield the best convergence.*

$8 \times 8 \times 8$. At each iteration, the norm of the residual is calculated and plottet over time, as shown in Figure 4. From the multigrid plots we clearly see that the number of restrictions is related to the rate of convergence, with the best improvement for the most restrictions. We have observed similar results for various grid resolutions, where only the GPU memory limits the grid sizes. In the following tests we use 25 iterations for the Jacobi method and two V-cycles of four restrictions, which requires roughly the same amount of time, as shown in Figure 4.

The previous test indicate that the multigrid method provides better accuracy, using the same time. In this next test we examine the impact on the divergence. Since the solution to the Poisson pressure equation is directly related to the preservation of the continuity constraint, the velocity divergence indicates how good the solution actually is. Figure 5 illustrates the divergence field at the center slice of the z-direction. Light grey represents positive divergence and dark grey represents negative divergence. The setup consists of two ventilators, one at the bottom left and one at the top right, a static obstacle in the middle and a smoke source at the bottom. Both illustrations indicate high-frequency disturbance, but only the Jacobi method suffers from low-frequency errors. This underlines the purpose of the multigrid method, namely to reduce the appearance of low-frequent errors.

Finally we examine whether the improvements yield better visual details. A visual representation of the same setup

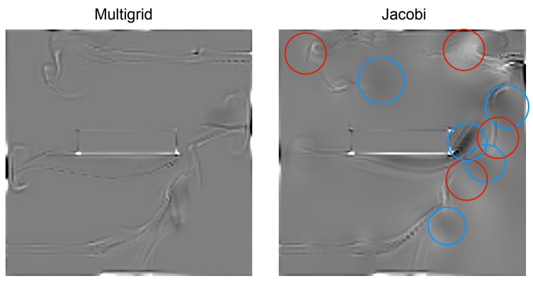

Figure 5: *Example of the velocity divergence from the center slice in the z-direction. The Jacobi method to the right suffers from low-frequent divergence, indicated by the circles.*

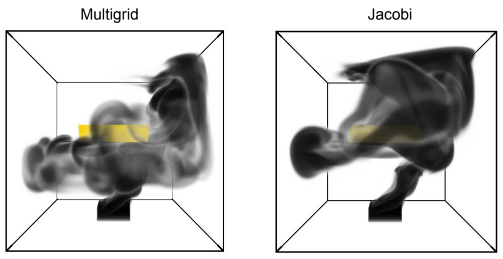

Figure 6: *Visual comparison of the Jacobi and multigrid methods. The multigrid method creates finer vortices an better details. The smoke is more diffuse in the Jacobi case.*

as before is illustrated in Figure 6. There is a general tendency, that the Jacobi method causes the smoke to be thicker and more diffuse, whereas the multigrid method generates finer circular movements. One can think of the Jacobi iterations as a diffusion process of the initial guess. When the process is stopped before total convergence, the solution is left in a diffusive state. Since the multigrid method converges faster, the pressure field contains less diffusive behavior.

3.2. False Diffusion

False diffusion is numerical inaccuracy, introduced by the advection term. Though diffusion is a natural part of the Navier-Stokes equation, diffusion caused by advection is not. To test for false diffusion we turn off true diffusion and add cold and hot ventilators to the left and bottom sides of a cube. The ventilators blow in the same diagonal direction and the opposite sides are open, see Figure 7.

If there was no false diffusion we would expect a sharp transition between the cold and hot fluid along the diagonal. In [BMF07] the authors show that advection of a quantity q,

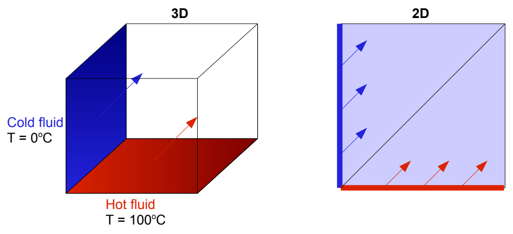

Figure 7: *Setup for the false diffusion test. Cold and hot ventilators blow in the diagonal direction. Mixing of temperatures along the diagonal will indicate the presence of false diffusion.*

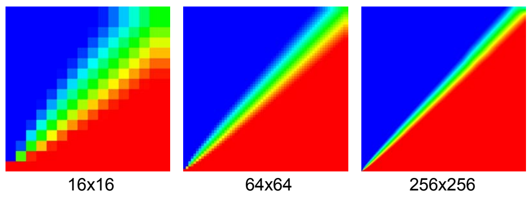

Figure 8: *False diffusion using three different grid resolutions. Semi-Lagrangian advection yields a clear relation between high resolutions and less false diffusion. Thus we expect false diffusion to influence low resolution simulations and should therefore use high resolutions when possible.*

using the Semi-Lagrangian approach equals,

$$\frac{\partial q}{\partial t} = -(u \cdot \nabla) q + u \cdot \Delta x \nabla^2 q. \tag{16}$$

This is advection plus an extra diffusion-like term, scaling with the grid cell sizes Δx. I.e. a fine grid yields less false diffusion than coarser grids. Figure 8 illustrates how our solution mixes the hot and cold temperatures due to false diffusion. As expected, false diffusion appears regardless of the resolutions, though better results are achieved for fine resolutions. Figure 9 shows a diagonal plot of the temperatures, which again concludes that false diffusion is most dominant in coarse resolutions. For future work, false diffusion could be minimized if the advection scheme was exchanged with a higher order accurate scheme, such as MacCormack [KC07] or QUICK [MCPN08].

3.3. Shear-Driven Cavity Flow

The cavity flow test includes separation and reattachment of the flow, which often appear in practical situations of engineering interest [Ben99,GGS82]. It is an important property of the solver, to reproduce these scenarios correctly. The cavity setup is illustrated in Figure 10.

One primary vortex should appear in the middle together with two secondary vortices in the lower corners. We

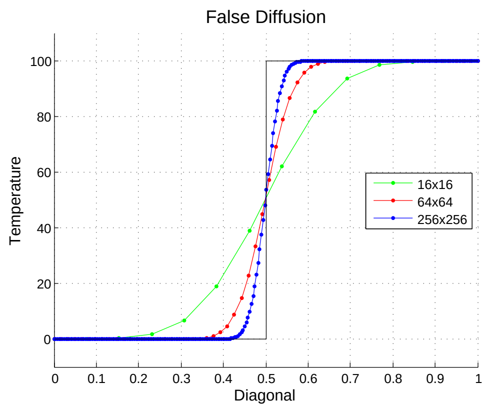

Figure 9: *Temperature profile measured perpendicular to the flow direction. The fine grid resolutions cause less false diffusion. The black line indicates the optimal solution.*

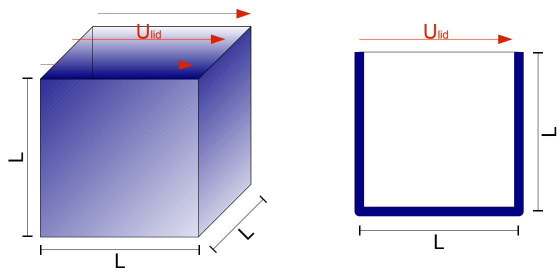

Figure 10: *A cubic cavity with a constant moving lid at velocity $u = [u_{lid}, 0, 0]^T$. The characteristic size of the cavity is L.*

will compare our results to the ones presented by Ghia et al. [GGS82], using a Reynolds number as,

$$\mathrm{Re} = \frac{u_{\text{lid}} L}{\nu} = 1000 \qquad (17)$$

where u_{lid} and L are the characteristic velocity and size of the cavity setup. The results in [GGS82] are all obtained from two dimensional simulations, so we apply free-slip conditions to the walls in the third direction, which should make the third dimension irrelevant. Figure 11 illustrates streamlines for the cavity flow using a $128 \times 128 \times 128$ grid. We obtain a flow that is very similar to the one presented in [GGS82]. For comparison, the sizes of the secondary vortices V^1 and V^2 are measured. Results are listed in Table 1.

The results summarized in Table 1 and the velocity profile measured along the vertical center line, illustrated in Figure 12, are quite good compared to the references. Resolutions of $128 \times 128 \times 128$ and above, yield very close results, whereas the lower resolutions suffer from smoothing.

Res.	64^3	128^3	256^3	Ghia et al. (129^2)
V_x^1	0.16	0.19	0.22	0.22
V_y^1	0.12	0.14	0.16	0.18
V_x^2	0.29	0.30	0.31	0.30
V_y^2	0.36	0.36	0.35	0.35

Table 1: *The x- and y-lengths of the two secondary vortices compared to [GGS82]. As the resolution increases the results become closer to the reference. All lengths are relative to L.*

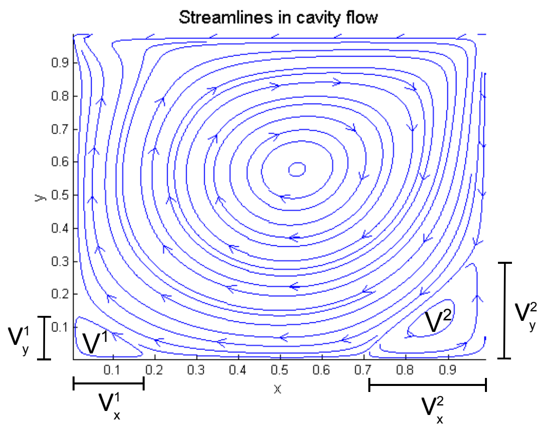

Figure 11: *Streamlines for the cavity flow, using a $128 \times 128 \times 128$ resolution. One primary vortex appears in the middle and two secondary vortices V^1 and V^2 in the bottom corners.*

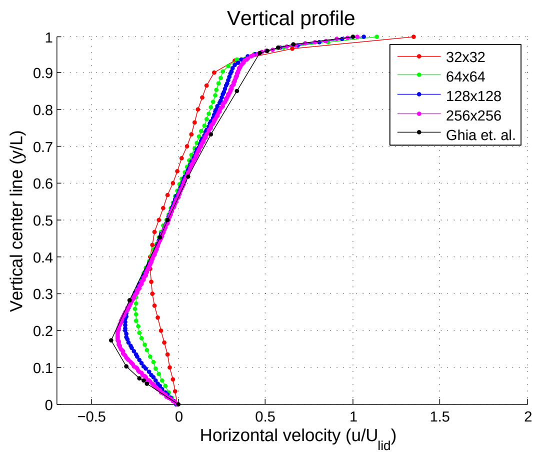

Figure 12: *The y-component velocity profile measured at the vertical center line. The high resolutions fit the reference well.*

As previously underlined, there is a strong connection between low resolutions and the numerical errors, caused by finite difference approximations.

4. Conclusion

We have combined the power of multiprocessor architectures on GPUs, with a fractional step method for solving the Navier-Stokes equations. The convergence rate of the Poisson solver has been improved with a multigrid solver and we presented our approach to handle internal boundaries at the coarse levels, without suffering from flow penetrations across boundaries. Our multigrid method compares well to referenced results of engineering standards.

The present solver is still in an early state and improvements are possible as future work. Dissipation caused by the spatial approximations are significant when resolutions are low, whereas higher resolutions minimize dissipation quite well. Thin boundaries limit the multigrid restrictions, due to collocated grids in combination with the current boundary conditioning approach. Fortunately, both of these issues can be avoided when the resolution is high, which the solver has shown to perform well at.

Rambøll Denmark A/S has already shown great interest, based on the intermediate results presented. Though accuracy is not as high as current CPU-based solvers, the present CUDA solver can be applied for fast and rough check and thereby evaluate new building designs quickly. Furthermore, the speed of the present solver makes it possible to analyse several different locations of fire sources, thereby improving and making buildings safer. Figure 13 illustrates an example of the time consumption for the solver currently used at Rambøll and our CUDA based solver. At resolutions of $128 \times 128 \times 128$ we have observed performance speedups of more than 350. The number of frames pr second for various grid resolutions are plotted in Figure 14.

GPU-based solutions, as the one we have just presented, open for the possibility of utilizing multiple GPUs. Thus, additional performance speedups are attainable with a limited amount of modifications and extra hardware. Currently we are investigating this topic, though no results are ready yet. Based on previous observations, we have found three primary targets for other future improvements.

- Incorporating a turbulence model, such as Large Eddy Simulation (LES) [RFBP97].
- Changing to a staggered grid and higher order difference schemes, to improve further on false diffusion.
- Using a multigrid method for the implicit diffusion equation.

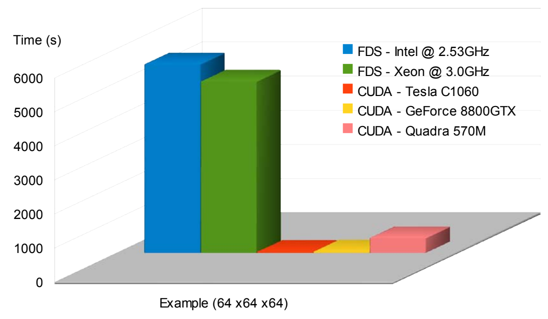

Figure 13: *Time comparison of FDS and the present CUDA solver for a 10 seconds simulation, using a 64^3 grid resolution. FDS requires nearly one and a half hour, our CUDA solver uses less than one minute on a modern GPU. We emphasize that the comparison is only to indicate production turn around time, and not a comparison of the solvers themselves.*

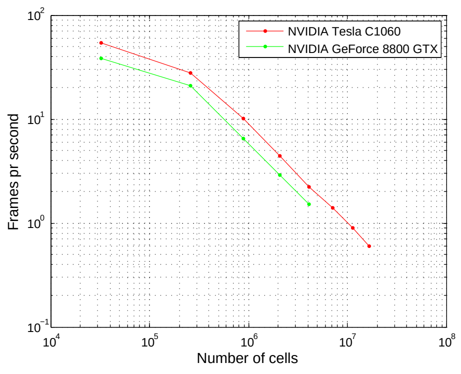

Figure 14: *Double logarithmic plot of frames pr second for various grid resolutions. The linear behavior indicates that frames pr second decrease as an inverse power function. Thus, when resolutions are high, there is a low penalty for further increasing the resolution.*

References

[Ben99] BENNETSEN J. C.: *Numerical Simulation of Turbulent Airflow in Livestock Buildings*. PhD thesis, The Department of Mathematical Modelling, The Technical University of Denmark, 1999.

[BHM00] BRIGGS W. L., HENSON V. E., MCCORMICK S. F.: *A Multigrid Tutorial*, 2 ed. SIAM: Society for Industrial and Applied Mathematics, July 2000.

[BMF07] BRIDSON R., MÜLLER-FISCHER M.: Fluid simulation: Siggraph 2007 course notes. In *SIGGRAPH '07: ACM SIGGRAPH 2007 courses* (New York, NY, USA, 2007), ACM, pp. 1–81.

[BP08] BRANDVIK T., PULLAN G.: Acceleration of a 3d euler solver using commodity graphics hardware. In *46th AIAA Aerospace Sciences Meeting and Exhibit* (January 2008), Whittle Laboratory, Department of Engineering, University of Cambridge.

[Bri08] BRIDSON R.: *Fluid Simulation*. A. K. Peters, Ltd., Natick, MA, USA, 2008.

[BT07] Becker M., Teschner M.: Weakly compressible sph for free surface flows. In *SCA '07: Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation* (Aire-la-Ville, Switzerland, Switzerland, 2007), Eurographics Association, pp. 209–217.

[ETK*08] Elcott S., Tong Y., Kanso E., Schröder P., Desbrun M.: Stable, circulation-preserving, simplicial fluids. In *SIGGRAPH Asia '08: ACM SIGGRAPH ASIA 2008 courses* (New York, NY, USA, 2008), ACM, pp. 1–11.

[fds09] FDS: Fire dynamics simulator. Online, June 2009. `http://www.fire.nist.gov/fds/`.

[FF01] Foster N., Fedkiw R.: Practical animation of liquids. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2001), ACM, pp. 23–30.

[FM96] Foster N., Metaxas D.: Realistic animation of liquids. *Graph. Models Image Process. 58*, 5 (1996), 471–483.

[FM97] Foster N., Metaxas D.: Modeling the motion of a hot, turbulent gas. 181–188.

[FP02] Ferziger J. H., Peric M.: *Computational Methods for Fluid Dynamics*, 3rd ed. Springer, 2002.

[FSJ01] Fedkiw R., Stam J., Jensen H. W.: Visual simulation of smoke. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2001), ACM, pp. 15–22.

[GGS82] Ghia U., Ghia K. N., Shin C. T.: High-re solutions for incompressible flow using the navier-stokes equations and a multigrid method. *Journal of Computational Physics 48* (December 1982), 387–411.

[Gli09] Glimberg S. L.: *Smoke Simulation for Fire Engineering using CUDA*. Master's thesis, Department of Computer Science, University of Copenhagen, June 2009.

[Har04] Harris M. J.: *Fast Fluid Dynamics Simulation on the GPU*. GPU Gems 1. Addison-Wesley, April 2004, ch. 38, pp. 637 – 665.

[HK05] Hong J.-M., Kim C.-H.: Discontinuous fluids. *ACM Trans. Graph. 24*, 3 (2005), 915–920.

[KC07] Keenan Crane Ignacio Llamas S. T.: *Real-Time Simulation and Rendering of 3D Fluids*. GPU Gems 3. Addison-Wesley, August 2007, ch. 30, pp. 633 – 675.

[KFCO06] Klingner B. M., Feldman B. E., Chentanez N., O'Brien J. F.: Fluid animation with dynamic meshes. *ACM Trans. Graph. 25*, 3 (2006), 820–825.

[LFWK06] Li W., Fan Z., Wei X., Kaufman A.: *Flow Simulation with Complex Boundaries*. GPU Gems 2. Addison-Wesley, December 2006, ch. 47, pp. 747 – 764.

[LGF04] Losasso F., Gibou F., Fedkiw R.: Simulating water and smoke with an octree data structure. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers* (New York, NY, USA, 2004), ACM, pp. 457–462.

[LSSF06] Losasso F., Shinar T., Selle A., Fedkiw R.: Multiple interacting liquids. *ACM Trans. Graph. 25*, 3 (2006), 812–819.

[MC09] Molemaker J., Cohen J. M.: A fast double precision cfd code using cuda. Extended abstract Presented at Parallel CFD, 2009.

[MCPN08] Molemaker J., Cohen J. M., Patel S., Noh J.: Low viscosity flow simulations for animation. In *SCA '08: Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2008), Eurographics Association, pp. 9–18.

[NVI09] NVIDIA: Nvidia compute unified device architecture. programming guide. *NVIDIA Corporation 2.2* (February 2009).

[RFBP97] Rodi W., Ferziger J. H., Breuer M., Pourquiée M.: Status of large eddy simulation: Results of a workshop. *Journal of Fluids Engineering 119*, 2 (1997), 248–262.

[RNGF03] Rasmussen N., Nguyen D. Q., Geiger W., Fedkiw R.: Smoke simulation for large scale phenomena. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers* (New York, NY, USA, 2003), ACM, pp. 703–707.

[Saa03] Saad Y.: *Iterative Methods for Sparse Linear Systems, 2nd edition*. SIAM, Philadelphia, PA, 2003.

[SFK*08] Selle A., Fedkiw R., Kim B., Liu Y., Rossignac J.: An unconditionally stable maccormack method. *J. Sci. Comput. 35*, 2-3 (2008), 350–371.

[Sta99] Stam J.: Stable fluids. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1999), ACM Press/Addison-Wesley Publishing Co., pp. 121–128.