A Mass Conserving Wind Model Evaluation
With Finite Element

Daniel Eriksson

Abstract

This work describes a method of approximating a wind field for an urban environment for the purpose of dispersion modelling. Instead of using the classic Navier Stokes equation, a mass conserving wind model is evaluated. The model uses an empirical diagnostic study to approximate a stationary wind field that is forced to be convergence free using a least square variational technique. This work has shown that there is a way to approximate the mass conserving wind field for a large urban environment using Comsol Multiphysics and the Finite Element Method. Compared to wind tunnel experiment the large features of the main flow are present but the wind speed is underestimated. Among the iterative solvers tested Multi grid and Conjugate Gradient performed best. An urban city with 2 100 000 degrees of freedom had a solution time of around three minutes.
Utvärdering av en massbevarande vindmodell löst med finita element

Sammanfattning
# Contents

1 Introduction .......................................................... 1
   1.1 Background ...................................................... 1
   1.2 Problem Description ........................................... 2
      1.2.1 Task ....................................................... 2
      1.2.2 Purpose .................................................. 2
      1.2.3 Restrictions ............................................. 2

2 Theory ............................................................... 3
   2.1 Wind models .................................................... 3
      2.1.1 Direct Numerical Simulation ............................. 3
      2.1.2 Turbulence models ...................................... 3
   2.2 Mass conservative model ...................................... 3
      2.2.1 Generating the initial field ............................ 4
      2.2.2 Adding recirculation zones ............................. 5
      2.2.3 Generating the mass conserving wind field .......... 6
      2.2.4 Boundary condition ..................................... 9
   2.3 Numerical method .............................................. 10
      2.3.1 The weak formulation .................................. 10
      2.3.2 The finite element approximation ...................... 11
      2.3.3 Assembling .............................................. 12

2.4 Solvers .......................................................... 14
   2.4.1 Relaxation Methods ...................................... 14
   2.4.2 Convergence of the relaxation method .................... 14
   2.4.3 Conjugate Gradient ...................................... 15
   2.4.4 Convergence of Conjugate Gradient ...................... 17
   2.4.5 MultiGrid ............................................... 18
   2.4.6 Preconditioner .......................................... 20

3 Method ............................................................ 21
   3.1 The Domain .................................................... 22
      3.1.1 The Grid .................................................. 23
   3.2 Solving ........................................................ 25
      3.2.1 Generating the initial field with recirculation zones 25
      3.2.2 Solving the Poisson equation .......................... 26
   3.3 Verification and testing ..................................... 27

4 Result .............................................................. 28
   4.1 Empty domain, part 1 ...................................... 28
   4.2 Block domain, part 2 ...................................... 29
   4.3 City, part 3 .................................................. 37

5 Discussion .......................................................... 39

6 Conclusion ........................................................ 40
   6.1 Future work ................................................... 40
## Appendix

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.1 Global minimum</td>
<td>41</td>
</tr>
<tr>
<td>A.2 Solution in n steps</td>
<td>42</td>
</tr>
<tr>
<td>A.3 Gram-Schmidt Conjugation</td>
<td>43</td>
</tr>
<tr>
<td>A.4 Only the previously residual</td>
<td>44</td>
</tr>
<tr>
<td>A.5 The computing time and error estimate for a laminar initial wind</td>
<td>45</td>
</tr>
<tr>
<td>profile</td>
<td></td>
</tr>
</tbody>
</table>
1 Introduction

This work is a master thesis describing an alternative way of approximating a stationary wind field in a complex and large urban domain. The work was established by the Swedish Defence Research Agent (FOI) together with Umeå University.

1.1 Background

Sweden together with Denmark, Norway, Poland and ten other countries worldwide are part of the crisis management system ARGOS. The system was first developed to support and help the Danish Emergency Management Agency (DEMA) with nuclear related incidents. Today ARGOS supports the respective countries emergency response organisations with decision making in matters concerning CBRN\textsuperscript{1} substances. Since 2005 the system is in use in Sweden and currently the Swedish Radiation Safety Authority (SSM) and FOI are utilizing the system. More information about ARGOS can be found in [2].

The need for a dispersion model for urban environments within ARGOS led to the task for FOI and the Danish Risø National Laboratory (DTU) to start developing the Nordic Urban Dispersion model (NUD). The assignment for FOI, given by SSM was to generate a wind-field, given actual meteorological wind information. Meanwhile DTU developed a dispersion model that used the wind field from FOI.

The emergency response application requires a short solution time, therefore a mass conserving wind model was chosen. The FOI report [1] illustrates the different techniques and recommendations for generating the wind field. The numerical method was Finite difference method (FDM). The iterative solvers were not able to solve the problem. Therefore, the model is currently implemented with a direct solver and to keep the short computing time demand it only handle small domains.

\textsuperscript{1}CBRN stands for Chemical (C), Biological (B), Radioactive (R) and Nuclear (N)
1.2 Problem Description

The assignment is to use Comsol Multiphysics ver. 4.3b (CM) as a testing platform to investigate if the model is solvable for a large complex domain. CM uses the Finite Element Method (FEM) and has a lot of different solvers that can be used.

1.2.1 Task

To investigate if the mass conservative wind-field model described in [1] can be solved with CM and FEM for large urban environments with a short solution time. If the model is solvable, comparison with experimental wind tunnel tests should be performed and convergence of different solvers should be tested.

1.2.2 Purpose

The work will be an example on how to solve the mass conservative model. The aim is to be able to make recommendations on which numerical discretization and iterative solver that are suitable to solve the problem. The report will form guidelines for future development of the dispersion model for urban environments within ARGOS.

1.2.3 Restrictions

This assignment will span from wind model theory to generating wind field through discretisation of a complex domain and many different solvers. Because of the extent of the task the focus will be to give a brief functional description on how to arrive at the mass conservative wind field with FEM. This means that the theory will be detailed up to a point but many interesting side tracks will be left unexplored.
2 Theory

This section will start by explaining fundamental theories of wind fields. Then the mass conserving wind model is explained followed by FEM. Finally different iterative solvers are explained, since large system of equations are generated.

2.1 Wind models

The classic model for describing a wind field is the Navier Stokes Equation (NSE). The NSE conserves momentum, mass and energy and describes the total motion of a fluid including turbulence. The derivation uses Newton’s second law of motion that states that “the rate of change of momentum of a fluid equals the sum of the external forces” [5]. The momentum is then expressed by the product of the acceleration and the mass. For a full derivation of the NSE see [5]. The incompressible NSE,

\[
\frac{du}{dt} + \hat{u} \cdot \nabla \hat{u} = -\frac{\nabla p}{\rho} + \nu \nabla^2 \hat{u} + \hat{g},
\]

\[\nabla \cdot \hat{u} = 0,
\]

where \(\hat{u}\) is the velocity vector, \(p\) is pressure, \(\rho\) is density, \(\nu\) is the kinematic viscosity and \(\hat{g}\) is the external force vector.

2.1.1 Direct Numerical Simulation

To be able to solve the complete spectrum of the flow with the NSE a Direct Numerical Simulation (DNS) is used. With that approach, the computational grid need to resolve the scales where the dissipation of energy take place according to the Kolmogorov theory from 1941. That means the grid is extremely fine and the computational cost is too high for DNS to be practical on a city size domain.

2.1.2 Turbulence models

There are techniques like Large Eddy simulation (LES) described in [4] and Reynolds Averaged Navier Stokes (RANS) that uses a coarser mesh and compensates the error by the use of turbulence models. These models are less computationally costly but when the domain reaches city size the computing time is too long for a fast dispersion model.

2.2 Mass conservative model

The the mass conservative model is described in [1], [3], [6], [7], [9] and [11]. Starting with the velocity \((V_{in})\) at a known altitude \((h_1)\) an initial wind field \((V_{00})\) is estimated in the domain with equations derived from meteorological studies. \(V_{00}\) is altered with recirculation zones based on layout and shapes of buildings. This gives an empirical wind field \((V_0)\). \(V_0\) is then forced to be convergence free using a least square variational technique described in [11]. This gives a mass conservative wind field \(V_u\). In figure 1 the steps are summarized.
2.2.1 Generating the initial field

The theory in this section is explained more in [3] and is originally a result from empirical studies. First a brief description of the basics of wind theory followed by the derivation of the approximated $\hat{V}_{00}$ field.

The geostrophic wind is directed along the isobars balanced by the pressure gradient force and the Coriolis force, with the high pressure on the right hand side in the northern hemisphere [10]. When the altitude decreases the wind is effected by ground friction and slows down, which decreases the effect of the Coriolis force and turn the wind to the left. In the NUD model [1] only the wind speed is effected by changing altitude.

The horizontal interpolation is described by a profile with three layers as seen in figure 2. The linear interpolation layer, the surface layer and the constant layer. The corresponding functions is described in equation (2).
\[ |V_{00}(z)| = \begin{cases} \left( \frac{|V_{in}(h_{i+1})|-|V_{in}(h_i)|}{h_{i+1}-h_i} \right) (z - h_i) + |V_{in}(h_i)| \quad \text{for } h_1 \leq z \leq h_i \\ \frac{|V_{in}(h_i)|}{\ln \left( \frac{z-d}{Z_0} \right)} \ln \left( \frac{z-d}{Z_0} \right) \quad \text{for } h_m \leq z \leq h_1 \\ \frac{|V_{in}(h_i)|}{\ln \left( \frac{h_1-d}{Z_0} \right)} \ln \left( \frac{h_1-d}{Z_0} \right) \quad \text{for } 0 \leq z \leq h_m \end{cases} \]\\ (2)

where \( h_1, h_2, \ldots, h_i \) represent the altitudes were \( V_{in}(h_i) \) is known, where \( i \) is the number of known wind-speed layers. From [3] \( d \) is 0.8 times the mean building height \( (h_m) \) described by equation (3) and \( Z_0 \) is described by equation (4).

\[ d = 0.8 \sum_i W_i L_i H_i \sum_i W_i L_i \]\\ (3)

\[ Z_0 = 0.2 \frac{\sum_i W_i L_i H_i}{L_x L_y} \]\\ (4)

where \( H, L, W \) are height, length and width of building \( i \). \( L_x \) and \( L_y \) are the dimensions of the domain. Equation (2) to (4) are designed so that \( |V_{00}(z)| \rightarrow 0 \) when \( z \rightarrow d + Z_0 \). The expression for \( \hat{V}_{00}(z) \) becomes,

\[ \hat{V}_{00}(z) = (u_{00}, v_{00}, w_{00}) = (|V_{00}(z)| \cos(\phi), -|V_{00}(z)| \sin(\phi), 0), \]\\ (5)

where \( \phi \) represents the wind direction.

### 2.2.2 Adding recirculation zones

Next step is to consider turbulence generated by obstacles within the domain. Due to the urban environment the ground is assumed flat except for buildings. The idea is to alter specific areas within the \( \hat{V}_{00} \) field so it behaves similarly to observed turbulence. This is done by modifying \( \hat{V}_{00} \) with recirculation zones giving \( \hat{V}_0 \). Buildings are assumed rectangular with length \( (L) \), height \( (H) \) and width \( (W) \).

![Figure 3: The three recirculation zones generated by a single building when the wind field is perpendicular.](image)

In [1], [3] and [11] the area in the proximity of a building is divided into three zones as seen in figure 3. When working with recirculation zones, \( X, Y \) and \( Z \) are the internal coordinate system used by a specific zone. The positions of the
coordinate systems are shown in figure 3.

The first zone is called the Displacement zone and is positioned upstream. The sum of the velocity vectors within the zone is zero and the zone reaches a length of \(L_f\) at ground level according to equation,

\[
L_f = \frac{2W}{1 + 0.8(W/H)}
\]  

(6)

The zone reaches a height of 0.6\(H\) on the upwind side of the building and the surface is defined by,

\[
\frac{X^2}{L_f^2(1 - (Z/0.6H)^2)} + \frac{Y^2}{W^2} = 1
\]

(7)

The second zone is called the Cavity zone and is positioned on the leeward side of the building. It expands downstream according to the surface,

\[
\frac{X^2}{L_r^2(1 - (Z/H)^2)} + \frac{Y^2}{W^2} = 1
\]

(8)

The length of this zone at ground level is \(L_r\) according to,

\[
L_r = \frac{1.8W}{(L/H)^{0.3}(1 + 0.24(W/H))}
\]

(9)

The velocity is in the opposite direction representing the lee-edge vortex and decays towards the ground according to equation (10) and (11).

\[
|\hat{V}_0(x,y,z)| = -\hat{V}_{00}(H) \left(1 - \frac{X}{d_n}\right)^2
\]

(10)

\[
d_n = L_r \sqrt{\left(1 - \left(\frac{Z}{H}\right)^2\right) \left(1 - \left(\frac{Y}{W}\right)^2\right)}
\]

(11)

The third zone is called Wake zone and reaches further downstream. The only difference compared to the Cavity zone is that the ground extension is three times \(L_r\) and that the velocity is described by,

\[
|\hat{V}_0(x,y,z)| = \hat{V}_{00}(Z) \left(1 - \frac{d_n}{X}\right)^{1.5}
\]

(12)

There is also described in [3] and [6] how to obtain the empirical parametrised field (\(\hat{V}_0\)) when the wind direction is not perpendicular with a building or when buildings are close to each other. These cases are excluded in this work.

2.2.3 Generating the mass conserving wind field

In this section the mass conserved field (\(\hat{V}_u\)) will be derived using a least square variational technique described in [11]. This procedure is used in [3], [6] and [7].

From the previous section the empirical parametrised field is \(\hat{V}_0 = (u_0, v_0, w_0)\).
By creating an adjusted field $\hat{V}_u = (u, v, w)$ with the condition that it should be as close as possible to $V_0$, the difference can be measured by the functional,  
\[
J[u, v, w] = \int \left( \alpha^2(u - u_0)^2 + \beta^2(v - v_0)^2 + \gamma^2(w - w_0)^2 \right) dV. \tag{13}
\]
where $\alpha$, $\beta$ and $\gamma$ are constants and controls the ratio between how close the different components of the $\hat{V}_u$ field will get to the $V_0$ field. The adjusted field should also be mass consistent which means that it should satisfy equation (14), derived in [12].

\[
G[u, v, z] = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = \nabla \cdot \hat{V}_u = 0 \tag{14}
\]
To get the adjusted field $(\hat{V}_u)$ to be both mass consistent and as close as possible to the empirical field $(\hat{V}_0)$ the Lagrangian multiplier method is used. The weak constraint of equation (13) is subjected to the strong constraint of equation (14) and by integrating over the whole domain give the specific functional shown in (15).

\[
E[u, v, w, \lambda] = \int_V \left( J[u, v, w] + \lambda G[u, v, w] \right) dV \tag{15}
\]
where $u, v, w$ are the adjusted field components in Cartesian coordinates and $u_0, v_0, w_0$ are the empirical field components. In [11] the horizontal weights, $\alpha$ and $\beta$ are equal which yields a horizontally rotationally symmetric model. $\lambda$ is the Lagrange Multiplier.

Taking the first variation with respect to $u$, $v$, $w$ and the Lagrange multiplier $\lambda$ and forcing the expressions to be zero a system of four equations is derived. The first variation with respect to $u$ is derived in equation (17) to (24).

\[
E[u_{\min} + \varepsilon \phi, v, w, \lambda] = P(\varepsilon), \tag{17}
\]
where $\varepsilon$ is a scalar and $\phi(x)$ is an arbitrary function with one derivative and satisfies certain boundary condition specified later. Here $\varepsilon \phi$ is called the variation of the function $u_{\min}$. The functional $E[u]$ is supposed to be minimised by $u = u_{\min}$ hence $P(\varepsilon)$ depends on $\varepsilon$ and has a minimum at $\varepsilon = 0$. The first variation with respect to $u$ can then be expressed by the derivative of $P(\varepsilon)$ at $\varepsilon = 0$.

\[
\partial_u E[\phi] = \frac{dP(\varepsilon)}{d\varepsilon} \bigg|_{\varepsilon=0} \tag{18}
\]
\[
= \lim_{\varepsilon \to 0} \int_V \left[ \frac{E[u_{\min} + \varepsilon \phi, v, w, \lambda] - E[u_{\min}, v, w, \lambda]}{\varepsilon} \right] dV \tag{19}
\]
\[
= \lim_{\varepsilon \to 0} \int_V \left[ \frac{\alpha^2(u_{\min} + \varepsilon \phi - u_0)^2 - \alpha^2(u_{\min} - u_0)^2 + \lambda d(u_{\min} + \varepsilon \phi) - \lambda d(u_{\min})}{\varepsilon} \right] dV
\]
Expanding the squares give,

\[
\partial_u E[\phi] = \lim_{\varepsilon \to 0} \int_V \left[ \frac{\alpha^2(2u_{\min} \varepsilon \phi + (\varepsilon \phi)^2 - 2\varepsilon \phi u_0) + \lambda (\partial \varepsilon \phi)^2}{\varepsilon} \right] dV \tag{20}
\]
Since $\varepsilon$ is a scalar it can be cancelled which give,
\[
\partial_u E[\phi] = \lim_{\varepsilon \to 0} \int_V \left[ \alpha^2(2u_{min}\phi + \varepsilon\phi^2 - 2\phi u_0) + \frac{d\phi}{dx} \right] dV \quad (21)
\]
\[
= \int_V \left[ \alpha^2(2u_{min}\phi - 2\phi u_0) + \lambda \frac{\partial \phi}{\partial x} \right] dV \quad (22)
\]
Using the product rule on the last term give,
\[
\partial_u E[\phi] = \int_V \left[ \alpha^2(2u_{min}\phi - 2\phi u_0) + \frac{\partial \lambda}{\partial x} \phi + \frac{\partial (\lambda \phi)}{\partial x} \right] dV \quad (23)
\]
Using the divergence theorem,
\[
\partial_u E[\phi] = \int \int \int \left[ \alpha^2(2u_{min}\phi - 2\phi u_0) - \frac{\partial \lambda}{\partial x} \phi \right] dV + \int \int \left[ \lambda \phi_n \right] dS \quad (24)
\]
Assuming that $\int \lambda \phi_n dS = 0$, the last term vanish. The effect of $\int \lambda \phi_n dS = 0$ will be addressed in section (2.2.4). Hence $\partial_u E[\phi] = 0$ gives,
\[
\int \int \int \left[ \alpha^2(2u_{min} - 2u_0) - \frac{\partial \lambda}{\partial x} \right] \phi dV = 0 \quad (25)
\]
The first Euler equation becomes,
\[
u_{min} = u_0 + \frac{1}{2\alpha^2} \frac{\partial \lambda}{\partial x} \quad (26)
\]
The same procedure can be used to derive expressions for $v_{min}$ and $w_{min}$,
\[
\begin{align*}
v_{min} &= v_0 + \frac{1}{\beta^2} \frac{\partial \lambda}{\partial y} \\
w_{min} &= w_0 + \frac{1}{\gamma^2} \frac{\partial \lambda}{\partial z}
\end{align*} \quad (27) \quad (28)
\]
Taking the first variation with respect to $\lambda$ and following the same procedure,
\[
\frac{\partial u_{min}}{\partial x} + \frac{\partial v_{min}}{\partial y} + \frac{\partial w_{min}}{\partial z} = 0 \quad (29)
\]
Substituting equation (26) to (28) into (29) gives the Poisson equation,
\[
\frac{1}{2\alpha^2} \frac{\partial^2 \lambda}{\partial x^2} + \frac{1}{2\beta^2} \frac{\partial^2 \lambda}{\partial y^2} + \frac{1}{2\gamma^2} \frac{\partial^2 \lambda}{\partial z^2} = - \left( \frac{\partial u_0}{\partial x} + \frac{\partial v_0}{\partial y} + \frac{\partial w_0}{\partial z} \right) \quad (30)
\]
In operator form the equation can be written as,
\[
\nabla \cdot \hat{c} \nabla \lambda = f \quad (31)
\]
where
\[
\hat{c} = [1/2\alpha^2, 1/2\beta^2, 1/2\gamma^2] \quad (32)
\]
and
\[
f = -\nabla \cdot \hat{V}_0 \quad (33)
\]
Solving this together with equations (26) to (28) the mass consistent field $\hat{V}_u$ can be derived.
2.2.4 Boundary condition

When the mass conserving field was derived three conditions were imposed for each velocity component. The conditions are,

\[ \int_{\partial V} \left[ \lambda \phi_x u_x n_x \right] dS = 0 \]  (34a)
\[ \int_{\partial V} \left[ \lambda \phi_y v_y n_y \right] dS = 0 \]  (34b)
\[ \int_{\partial V} \left[ \lambda \phi_z w_z n_z \right] dS = 0 \]  (34c)

The boundary condition for \( \lambda \) need to satisfy equations (34a - 34c). At the same time must the mass conserving wind field satisfy \( \hat{n} \cdot \hat{V}_u = 0 \) on the wall boundary. First the wall boundary condition is explained followed by the atmosphere (free) boundary condition.

On the wall boundary \( \hat{V}_0 \neq 0 \) and after \( \hat{V}_u \) has been generated the condition \( \hat{n} \cdot \hat{V}_u = 0 \) should hold. The Euler equations were derived by using equations (34a - 34c), so by using them a satisfied boundary condition can be expressed. From equation (26) to (28) the boundary condition becomes,

\[ \hat{n} \cdot \nabla \lambda = 2\alpha^2 (u_{min} - u_0) n_x + 2\beta^2 (v_{min} - v_0) n_y + 2\gamma^2 (w_{min} - w_0) n_z \]  (35)

On this boundary \( \hat{n} \cdot \hat{V}_0 \neq 0 \) and \( \hat{n} \cdot \hat{V}_u = 0 \). That means the boundary condition become \( \hat{n} \cdot \hat{c} \nabla \lambda = -(u_0 + v_0 + w_0) \cdot \hat{n} \). Where \( \hat{c} \) is according to equation (32). This is consistent with the condition \( \phi = 0 \) in the variational formulation.

On the free boundary \( \hat{V}_u \) can vary, meaning mass should be able to leave or enter the domain. Instead of using the Euler equations to satisfy the conditions given by equations (34a - 34c), \( \lambda = 0 \) is used. The fact that \( \lambda \) is zero does not imply that \( \partial \lambda / \partial n = 0 \). So from equation (26) to (28) \( \hat{n} \cdot \hat{V}_u \) is probably not the same as \( \hat{n} \cdot \hat{V}_0 \) representing a movement of mass through the boundary.
2.3 Numerical method

The $\hat{V}_u$ field is derived by solving the Poisson equation. There are many methods that can be used to solve this equation within a domain. Methods like the Finite Different Method (FDM) and the Finite Volume Method (FVM). CM uses Finite Element Method (FEM) described in [5] and [13]. The idea is to discretise the domain and approximate the solution by continuous piecewise polynomials with the Galerkin approximation method. In this section the FEM is described.

2.3.1 The weak formulation

The problem can be formulated in the following way: find $\lambda$ such that,

\[-\nabla \cdot (\hat{c} \nabla \lambda) = f, \quad \text{in } \Omega \quad (36)\]
\[\hat{n} \cdot (\hat{c} \nabla \lambda) = k, \quad \text{in } \partial \Omega_1 \quad (37)\]
\[\lambda = 0, \quad \text{in } \partial \Omega_2 \quad (38)\]

where $\hat{c}$ is defined in equation (32),

\[f = (\nabla \cdot V_0), \quad k = -(u_0 + v_0 + w_0) \hat{n}, \quad \nabla = (\partial/\partial x, \partial/\partial y, \partial/\partial z),\]

$\omega$ is the complete domain, $\partial \omega_1$ is the wall boundary and $\partial \omega_2$ is the free boundary. First define where to find the solution. The space is defined as,

\[H^1(\Omega) = \{\lambda \in L_2(\Omega), \nabla \lambda \in L_2(\Omega)\} \quad (39)\]

where $L_2(\Omega) = \{\lambda : \Omega \rightarrow \mathbb{R}^3, ||\lambda|| < \infty\}$ and $||\lambda|| = (\int_{\Omega} |\lambda|^2 \partial \Omega)^{1/2}$. The space of test function $v$ is defined as,

\[H^1_0(\Omega) = \{v \in H^1(\Omega) | v_{|\partial \omega_2} = 0\} \quad (40)\]

Multiplying the first equation in (38) with the test function and integrating with Green’s formula,

\[\int_{\Omega} [fv] dV = -\int_{\Omega} [\hat{c} \nabla \lambda v] dV \quad (41)\]
\[= \int_{\Omega} [\hat{c} \nabla \lambda \cdot \nabla v] dV - \int_{\partial \Omega_1} [\hat{c} \nabla \lambda v \cdot n] dS \quad (42)\]
\[= \int_{\Omega} [\hat{c} \nabla \lambda \cdot \nabla v] dV - \int_{\partial \Omega_1} [kv] dS - \int_{\partial \Omega_2} [\hat{c} \nabla \lambda v \cdot n] dS \quad (43)\]
\[= \int_{\Omega} [\hat{c} \nabla \lambda \cdot \nabla v] dV - \int_{\partial \Omega_1} [kv] dS \quad (44)\]

The weak formulation becomes: Find $\lambda \in H^1(\Omega)$ such that $\lambda = g$ on $\partial \Omega_2$ and,

\[\int_{\Omega} [fv] dV = \int_{\Omega} [\hat{c} \nabla \lambda \cdot \nabla v] dV \quad (45)\]

Here the essential boundary condition is defined by construction of the solution space as $g$ and in the test space as 0. The natural boundary condition is defined with the weak formulation.
2.3.2 The finite element approximation

The finite element approximation is obtained as a linear combination of basis functions. The basis functions are constructed in the following way:

We suppose that the domain is subdivided into a finite number of tetrahedrons. Bricks can also be used, this is explained more in [5]. On each tetrahedral, the basis functions are linear and defined as,

\[ \varphi_j(p_i) = \begin{cases} 
1 & \text{for } i = j \\
0 & \text{for } i \neq j 
\end{cases} \quad (46) \]

where \( p_i \) is the position at node \( i \). Lets define a space \( V_h \) for the function \( \lambda_h \),

\[ V_h(\Omega) = \{ \lambda_h \in C(\Omega) \mid \lambda_h|_T \in F, \forall T \in T_h \} \quad (47) \]

where \( C(\Omega) \) is the space of all continuous functions, \( F = \{ a_0 + a_1x + a_2y + a_3z \mid a_0, a_1, a_2, a_3 \in \mathbb{R} \} \), \( T \) is a tetrahedral element and \( T_h \) are the vertices creating the tetrahedral mesh. A quadratic approximation can also be made, more about quadratic polynomials can be read in [5]. Function \( \lambda_h \) for the domain \( \Omega \) can now be expressed as,

\[ \lambda_h(x, y, z) = \sum_{j=1}^{n} \lambda(p_j)\varphi_j(x, y, z) \quad (48) \]

where \( n \) is the number of vertices in the domain \( \Omega \). The boundary conditions are both natural (Neumann) and essential (Dirichlet). By identifying the Dirichlet vertexes the function \( \lambda_h \) can be rewritten as,

\[ \lambda_h(x, y, z) = \sum_{j=1}^{m} \lambda(p_j)\varphi_j(x, y, z) + \sum_{j=m+1}^{n} \lambda(p_j)\varphi_j(x, y, z) \quad (49) \]

where \( (n-m) \) is the number of boundary nodes with a Dirichlet condition. The solution space and test space are,

\[ V_{h,1}(\Omega) = \{ \lambda_h \in V_h(\Omega) \mid \lambda_h|_{\partial\Omega_2} = g \} \quad (50) \]
\[ V_{h,0}(\Omega) = \{ v_h \in V_h(\Omega) \mid v_h|_{\partial\Omega_2} = 0 \} \quad (51) \]

Using the approximation \( \lambda \approx \lambda_h, f \approx f_h, k \approx k_h \) and \( v \approx v_h \), The Finite Element Method: Find \( \lambda_h \in V_{h,1} \subset H^1 \) such that,

\[ \int_{\Omega} [f_h v_h] \, dV = \int_{\Omega} [\hat{c}_j \nabla \lambda_h \cdot \nabla v_h] \, dV - \int_{\partial\Omega_1} [k_h v_h] \, dS \quad \forall v_h \in V_{h,0} \quad (52) \]

The Galerkin system,

\[ \int_{\Omega} f_j \varphi_j \, \varphi_i \, dV + \int_{\partial\Omega_1} k_{j} \varphi_j \, \varphi_i \, dS = \int_{\Omega} \left( \sum_{j=1}^{m} \lambda_j \hat{c}_j \varphi_j + \sum_{j=m+1}^{n} \lambda_j \hat{c}_j \varphi_j \right) \cdot \nabla \varphi_i \, dV \quad (53) \]
Rearranging,
\[
\int_{\Omega} f_j \varphi_j \varphi_i \, dV + \int_{\partial \Omega_1} k_j \varphi_j \varphi_i \, dS = \sum_{j=1}^{m} \lambda_j \int_{\Omega} \nabla \varphi_j \cdot \nabla \varphi_i \, dV + \sum_{j=m+1}^{n} \lambda_j \int_{\Omega} \nabla \varphi_j \cdot \nabla \varphi_i \, dV
\]
(54)
Because \( j = m + 1, \ldots, n \) represent the Dirichlet boundary \( d\Omega_2 \) and equation (38) puts \( g = 0 \) in \( V_{h,1} \) the last terms vanish.

\[
\int_{\Omega} f_j \varphi_j \varphi_i \, dV + \int_{\partial \Omega_1} k_j \varphi_j \varphi_i \, dS = \sum_{j=1}^{m} \lambda_j \int_{\Omega} \nabla \varphi_j \cdot \nabla \varphi_i \, dV
\]
(55)

Rearranging
\[
\sum_{j=1}^{m} \left( \int_{\Omega} \nabla \varphi_j \cdot \nabla \varphi_i \right) \lambda_j \, dV = \int_{\Omega} f_j \varphi_j \varphi_i \, dV + \int_{\partial \Omega_1} k_j \varphi_j \varphi_i \, dS
\]
(56)

Introducing the notations,
\[
A_{n \times n} = [a_{i,j}], \quad a_{i,j} = \int_{\Omega} \nabla \varphi_j \cdot \nabla \varphi_i \, dV
\]
\( j = 1, 2, 3, \ldots, m \)
\[
a_{i,j} = 0
\]
\( j = m + 1, \ldots, n \)
\[
F_{n \times n} = [F_{i,j}], \quad F_{i,j} = \int_{\Omega} \varphi_j \varphi_i \, dV
\]
\[
K_{n \times n} = [K_{i,j}], \quad K_{i,j} = \int_{\partial \Omega_1} \varphi_j \varphi_i \, dS
\]
where \( i, j = 1, 2, 3, \ldots, n \). Above two mass matrices are defined. The matrix \( K \) is only integrated over part of the boundary.

\[
\hat{b}_{n \times 1} = [b_j], \quad b_j = \left( \sum_{i=1}^{n} F_{i,j} f_i + \sum_{i,p \in \Omega_1} K_{i,j} k_i \right)
\]
(60)

where \( j = 1, 2, 3, \ldots, n \). we get,
\[
\sum_{j=1}^{n} a_{i,j} \lambda_j = b_j \quad \text{for} \quad i = 1, 2, \ldots, n
\]
(61)

Here \( a_{i,j} \) is an element in the stiffness matrix and \( b_j \) an element in the load vector.

2.3.3 Assembling

The assembling of the stiffness matrix \( A \) will be explained in this section. For the load vector see [5] and [13]. Lets consider each element separately. Instead of integrating over the whole domain we are summing the contributions from each tetrahedral.

\[
A = \sum_{j=1}^{m} \left( \sum_{k=1}^{t} \int_{T_k} \nabla \varphi_j \cdot \nabla \varphi_i \, dV \right) \quad \text{for} \quad i = 1, 2, \ldots, n
\]
(62)

12
were $T_k$ is a specific tetrahedral and $l$ is the total number of tetrahedrals in $\Omega$. On any particular element $T_k$ there is only four basis functions that are non zero and we only need to consider the case when both $\varphi_i$ and $\varphi_j$ are non zero. We can then create a local stiffness matrix associated with element $T_k$ with the dimensions $4 \times 4$. If the four nodes in $T_k$ are positioned at $P_s$ where $s = 1, 2, 3, 4$. A linear equation system for $\varphi_1$ looks like,

$$
\begin{bmatrix}
1 \\
0 \\
0 \\
0
\end{bmatrix} =
\begin{bmatrix}
\varphi_1(P_1) \\
\varphi_1(P_2) \\
\varphi_1(P_3) \\
\varphi_1(P_4)
\end{bmatrix} =
\begin{bmatrix}
x_1 & y_1 & z_1 \\
x_2 & y_2 & z_2 \\
x_3 & y_3 & z_3 \\
x_4 & y_4 & z_4
\end{bmatrix}
\begin{bmatrix}
a_1 \\
b_1 \\
c_1 \\
d_1
\end{bmatrix}
\tag{63}
$$

That means $\varphi_1 = a_1 + xb_1 + yd_1 + ze_1$ and $\nabla \varphi_j = (b_1, d_1, e_1)$ for a linear approximation. For a quadratic approximation see [5]. Doing the same procedure for $\varphi_2, \varphi_3$ and $\varphi_4$ gives,

$$
\nabla \varphi_s = (b_s, d_s, e_s) \text{ for } s = 1, 2, 3, 4
\tag{64}
$$

Because equation (64) only contains constants the integral can be expressed as the volume of the tetrahedral. The local stiffness matrix becomes,

$$
A_{s,t}^{T_k} = (c_1b_1b_t + c_2d_sd_t + c_3e_se_t)|T_k| \text{ for } s,t = 1, 2, 3, 4
\tag{65}
$$

were $|T_k|$ is the volume. Each element in the local stiffness matrix $s,t$ correspond to a specific element $i,j$ in the stiffness matrix. The mapping procedure is found in [5] and [13].
2.4 Solvers

The theory in this section is written from [5],[14] and [15] and the focus lies on different iterative solvers with a stationary problem. Jacobi and Successive Over Relaxation (SOR) will be explained first and then Steepest descend, Conjugate Gradient (CG) and finally Multi Grid (MG). From the FEM formulation the linear equation system $Ax = b$ is sparse, positive definite ($x^TAx > 0$) and symmetric ($A^T = A$). These properties are important when understanding the underlying theory and will be used frequently in this section.

2.4.1 Relaxation Methods

The matrix $A$ can be defined as $A = L + D + U$ were $A, L, D, U \in \mathbb{R}^{n \times n}$. $L$ is the lower triangular part of $A$, i.e. a null matrix with the lower left off diagonal elements replaced with the elements from $A$. $U$ is called the upper triangular part of $A$ and $D$ is the diagonal part of $A$. The equation system can be written as,

$$Ax = b$$

$$(L + D + U)x = b \quad (66)$$

The different ways of splitting $A$ will give different types of iterative methods. By splitting $A$ as $D + (L + U)$ the Jacobi method is derived.

$$(D + (L + U))x = b$$

$$Dx = -(L + U)x + b$$

$$x = -D^{-1}(L + U)x + D^{-1}b$$

$$x = E_{jac}x + z_{jac} \quad (67)$$

Where $E_{jac} = -D^{-1}(L + U)$ and $z_{jac} = D^{-1}b$. The SOR method is derived by setting $wA = w(L + D + U)$.

$$w(D + L + U)x = wb$$

$$wDx + wLx + wUx = wb + Dx - Dx$$

$$(D + wL)x = wb - wDx - wUx + Dx$$

$$(D + wL)x = wb - (wD + wU - D)x$$

$$(D + wL)x = wb - (wU + (w - 1)D)x$$

$$x = (D + wL)^{-1}wb - (D + wL)^{-1}(wU + (w - 1)D)x$$

$$x = E_{sor}x + z_{sor} \quad (68)$$

Where $E_{sor} = -(D + wL)^{-1}(wU + (w - 1)D)$ and $z_{sor} = (D + wL)^{-1}wb$. By defining $x_{i+1} = f(x_i)$ equation 67 can be written as,

$$x_{i+1} = E_{jac}x_i + z_{jac} \quad (69)$$

The equation can hopefully be solved by giving $x_0$ a value and substituting $x_i$ with $x_{i+1}$ until $x_{i+1} \approx x_i$. The same procedure is used for SOR.

2.4.2 Convergence of the relaxation method

After each iteration $x_{i+1}$ needs to be closer to the solution $x$. Lets define $x_n = x + \epsilon_n$, where $\epsilon_n$ is the error at step n. So after each iteration $||\epsilon_{i+1}||$ needs to be smaller than $||\epsilon_i||$ for the method to converge. Substituting $x_i = x + \epsilon_i$ into equation (69) give,
\[ x_{i+1} = E_{jac}(x + \epsilon_i) + z_{jac} \]
\[ = E_{jac}x + z_{jac} + E_{jac}\epsilon_i \]
\[ = x + E_{jac}\epsilon_i \]

\[ x_{i+1} - x = E_{jac}\epsilon_i \]
\[ \epsilon_{i+1} = E_{jac}\epsilon_i \] (70)

This means that the convergence depends on \( E_{jac} \). Let’s consider the right hand side of equation (70). Because \( E_{jac} \) is equivalent with \( E_{sor} \) and \( \epsilon_i \) is equivalent with \( \epsilon_{i+1} \) at this point, they will be called \( E \) and \( \epsilon \). Assume that \( E \) is symmetric and write \( \epsilon \) as a linear combination of the \( n \) eigenvectors \( v_i \) of \( E \). That give us,
\[ \epsilon = a_1 v_1 + a_2 v_2 + ... + a_n v_n \] (71)

Multiplying with \( E^k \) from the left give,
\[ E^k \epsilon = a_1 E^k v_1 + a_2 E^k v_2 + ... + a_n E^k v_n \] (72)

Here \( k \) represents the number of iterations or the number of times a new error is calculated. The definition of eigenvalue is \( Ev = lv \), where \( l \) is an eigenvector. This means an eigenvector does not change direction besides in the complete opposite direction when it’s pre-multiplied by \( E \). Equation (72) is written as,
\[ E^k \epsilon = a_1 l_1^k v_1 + a_2 l_2^k v_2 + ... + a_n l_n^k v_n \] (73)

If the largest eigenvalue is less then 1 the expression will converge to 0 if \( k \) goes to infinity. If \( E \) also is positive definite the eigenvalues are positive. That means that \( \max(l_i) < 1 \) is required for convergence. Its mentioned in [15] that there are non symmetric matrices that are called defective that does not have linear independent eigenvectors but still converges if \( \max|l_i| < 1 \).

2.4.3 Conjugate Gradient
To understand CG the methods Steepest descent will be explained first. Starting with \( Ax = b \). Let’s define the function
\[ f(x) = \frac{1}{2} x^T Ax - b^T x \] (74)
Calculating the gradient,
\[ \nabla f(x) = Ax - b = -r \] (75)

Minimizing the function by putting \( \nabla f(x) = 0 \) gives \( Ax = b \). This means \( Ax = b \) can be solved by finding \( x \) that minimizes \( f(x) \). Because matrix \( A \) is symmetric and positive definite it is shown (Appendix A.1) that their is only one global minimum. The gradient \( \nabla f(x) \) is a vector that points in the direction of the greatest increase of \( f(x) \). That means the solution lies somewhere in the opposite direction. The idea is to travel along the line \( -\nabla f(x) \) until the minimum is found along that line. Then continue with the same procedure until we arrive at the solution. If \( \alpha \) is the distance to travel the iteration becomes,
\[ x_{i+1} = x_i - \alpha \nabla f(x_i) = x_i + \alpha r_i \] (76)
where $\alpha$ is the distance to the minimum along the direction $r_i$. $x_{i+1}$ is now obtained by minimizing $f(x_i + \alpha r_i)$ with respect to $\alpha$, 

$$\frac{\partial}{\partial \alpha} f(x_i + \alpha r_i) = \nabla f(x_i + \alpha r_i)^T r_i = 0$$  \hspace{1cm} (77)

This means $\nabla f(x_{i+1}) \perp r_i$ as seen in figure 4.

![Figure 4: Typical path taken by the steepest descent method.](image)

An expression for $\alpha$ can be derived by equation (75) and (77).

$$0 = \nabla f(x_i + \alpha r_i)^T r_i = (A(x_i + \alpha r_i) - b)^T r_i = -r_i^T r_i + \alpha r_i^T Ar_i$$  \hspace{1cm} (78)

which gives,

$$\alpha_i = \frac{r_i^T r_i}{r_i^T Ar_i}$$  \hspace{1cm} (79)

The method for Steepest descent becomes,

$$r_i = b - Ax_i$$  \hspace{1cm} (80)

$$\alpha_i = \frac{r_i^T r_i}{r_i^T Ar_i}$$  \hspace{1cm} (81)

$$x_{i+1} = x_i + \alpha_i r_i$$  \hspace{1cm} (82)

With this iteration the direction of the search is only towards the solution if $f(x)$ is circular. Steepest descent have no restriction for using the same search direction over and over again. Let’s instead define $x_{i+1} = x_i + \alpha_i d_i$. Where $d_i$ is the new search direction and is part of a set of $A$ orthogonal vectors. If two vectors are orthogonal the inner product is zero ($\langle a, b \rangle = a^T b = 0$). Then if the vectors are $A$ orthogonal $\langle a, Ab \rangle = a^T Ab = 0$ and $a \perp Ab$.

First rewrite the expression for $-\nabla f(x_i)$,

$$-\nabla f(x_i) = r_i = b - Ax_i = b - A(x + e_i) = b - Ax - Ae_i = -Ae_i$$  \hspace{1cm} (83)
Then minimising $f(x_{i+1})$,
\[
\frac{d}{d\alpha} f(x_i + \alpha d_i) = \nabla f(x_i + \alpha d_i)^T d_i = (A(x_i + \alpha d_i) - b)^T d_i = (-r_i + \alpha Ad_i)^T d_i
\]  
which gives,
\[
\alpha_i = \frac{d_i^T r_i}{d_i^T Ad_i} \quad (85)
\]
\[
= -\frac{d_i^T Ae_i}{d_i^T Ad_i} (-Ae_i = r_i) \quad (86)
\]
By using this expression for $\alpha_i$ the search direction become $A$ orthogonal to the previous. Assume also that $d_i^T Ad_j = 0$ for all $i \neq j$. That means each search direction is only used one time. Because matrix $A$ has finite dimensions and each search direction is $A$ orthogonal to the rest it only takes $n$ search directions to arrive at the solution (Appendix A.2). The requirement for the search direction can be achieved by the Gram-Schmidt Conjugation (Appendix A.3). The expression for the search direction becomes,
\[
d_i = r_i + \sum_{j=0}^{i-1} \beta_{ij} d_j \quad (87)
\]
were,
\[
\beta_{ij} = -\frac{r_i^T Ad_j}{d_j^T Ad_j} \quad (j < i) \quad (88)
\]
At this point all the previous search directions need to be saved to construct $\beta_{ij}$. In Appendix A.4 it is shown that there is no need to save previous search directions and equation (88) can be expressed as,
\[
\beta_i \equiv \beta_{i,i-1} = \frac{r_i^T r_i}{r_{i-1}^T r_{i-1}} \quad (89)
\]
The complete CG method can be written as,
\[
1 \quad d_0 = r_0 = b - Ax_0, \quad \text{(By Equation 83.)} \quad (90)
2 \quad \alpha_i = \frac{d_i^T r_i}{d_i^T Ad_i}, \quad \text{(By Equation 86.)} \quad (91)
3 \quad x_{i+1} = x_i + \alpha_i d_i \quad (92)
4 \quad r_{i+1} = r_i - \alpha_i Ad_i, \quad \text{(By Equation 123.)} \quad (93)
5 \quad \beta_i = \frac{r_i^T r_i}{r_{i-1}^T r_{i-1}} \quad \text{(By Equation 128.)} \quad (94)
6 \quad d_{i+1} = r_{i+1} + \beta_{i+1} d_i, \quad \text{(By Equation 118.)} \quad (95)
\]
Here only equation (90) is used on the first iteration. Thereafter equation (91) to (95) is repeated until $(r_{i+1})$ is sufficient small. In [5],[14] and [15] more can be read about CG.

2.4.4 Convergence of Conjugate Gradient

The previous section shows that the CG method imposes two conditions on matrix $A$. The matrix needs to be both symmetric and positive definite for
the theory to hold. As in the Jacobi and SOR methods the eigenvalues play a role in the convergence. In equation (96), derived in [15] a bound describes the relation between the eigenvalues and the error.

\[ ||x - x_n||_A \leq 2 \left[ \frac{\sqrt{K} - 1}{\sqrt{K} + 1} \right]^n ||x - x_0|| \]  
\[ (96) \]

On the L.H side is the A-norm of \( x - x_n \) after \( n \) iteration. On the R.H side is the A-norm of \( x - x_0 \) multiplied by a factor. For the problem to converge this factor needs to be \(< 1\). \( K \) is the spectral condition number defined as \( l_{max}/l_{min} \).

### 2.4.5 MultiGrid

The theory in this section is explained deeper in [14] and [5]. The idea behind MultiGrid (MG) is to help the relaxation method to converge by switching between different mesh sizes. When a relaxation solver is running the convergence rate often have a tendency to slow down. In [14] this is explained by the ability to damp the low-frequency modes. The high-frequency modes are damped first increasing the convergence rate initially. By going to a coarser mesh the low frequency modes act like high frequency modes and can be damped by the solver. There are two different types of MG. The one explained in this report is the Geometric MG. There are also a Algebraic MG that do not need any information about the geometry.

The MG solver can be built in many different ways. But there is basically one method performed over and over again between different mesh sizes. This step is explained below.

First we derive an expression for the residual,

\[ r = b - A\tilde{x} = b - A(x - e) = b - Ax + Ae = Ae \]  
\[ (97) \]

where \( \tilde{x} \) is the approximated solution and \( e \) is the error.

First a relaxation solver like the Jacobi or the SOR is used with a fixed number of iterations to damp the high frequency. Then the residual is calculated and an operator maps the residual onto the coarser grid.

\[ r_{2h} = Rr_h \]  
\[ (98) \]

Here \( r_{2h} \) is the residual corresponding to the coarser mesh, \( R = \frac{1}{2}(I_{2h})^T \) is the restriction operator, and \( r_h \) is the residual corresponding to the finer mesh. The restriction operator will be explained with a one dimensional example.

![Figure 5: Shows two different mesh sizes. v represent the function values on the fine grid and u the coarser grid.](image-url)
If \( n \) is an odd number the grid size can be changed from \( h \rightarrow 2h \) as seen in figure 5. The idea is to represent the residual on a coarser grid. At this point we have an approximated solution in the domain. This solution could be represented on a coarser grid by a linear approximation based on the distance between the grid points. The same idea is used to represent the residual on the coarse grid.

\[
\begin{aligned}
    u_i &= \frac{(v_{2i}/2+v_{2i+1}+v_{2i+2}/2)}{2} \quad (i = 0, 1, \ldots, \frac{n+1}{2} - 1)
\end{aligned}
\]

(99)

The easiest and most common way of constructing the matrix is to go through it and add values around each position where a node has the same position on both grids. From (99) the vector becomes \([1, 2, 1, \ldots, 1, 2, 1]\) were the center value lie on the specific node. The restriction operator becomes,

\[
\begin{pmatrix}
    1 & 2 & 1 \\
    1 & 2 & 1 \\
    \vdots & & \ddots \\
    1 & 2 & 1
\end{pmatrix}
\begin{bmatrix}
    v_0 \\
    v_1 \\
    v_2 \\
    v_3 \\
    \vdots \\
    v_n
\end{bmatrix}
= \begin{bmatrix}
    u_0 \\
    u_1 \\
    u_2 \\
    u_3 \\
    \vdots \\
    u_{n+1}
\end{bmatrix}
\]

(100)

\[
k_h(I_{2h}^h)^T \hat{v} = \hat{u}
\]

(101)

Instead of using \( \hat{v} \) and \( \hat{u} \) the residual \( r_h \) from the finer grid \( h \) is used. The equation system \( A_{2h}e_{2h} = r_{2h} \) is iterated a fixed number of times. To solve \( A_{2h}e_{2h} = r_{2h} \) we need to know \( A_{2h} \).

\[
\begin{aligned}
    A_{2h}e_{2h} &= r_{2h} \\
    &= Rr \\
    &= RAc \\
    &= RAPe_{2h} \\
    A_{2h} &= RAP \\
    A_{2h} &= \frac{3}{4}(I_{2h}^h)^T A I_{2h}^h
\end{aligned}
\]

(102)

The error \( e_{2h} \) is returned to the fine grid with,

\[
P e_{2h} = e_h
\]

(103)

Here \( P = \frac{1}{2}I_{2h}^h \) is called the prolongation operator. Back on the fine grid the error \( e_h \) can be used to alter \( \bar{x} \) and remove the part of the low frequency modes according to equation (104)

\[
x_{h, \text{new}} = x_h + e_h
\]

(104)

This new approximation can be iterated a few more times until the tolerance is satisfied. The procedure explained so far in this section is called a Two-Grid Cycle (TGC). The cycle can be used either to go to a finer grid or to a coarser grid. This means that you always have the possibility to go up or down in mesh size. The Two Grid cycle uses \( h \rightarrow 2h \rightarrow h \) but you can just as well use \( h \rightarrow 2h \rightarrow 4h \rightarrow 2h \rightarrow h \).

Instead of starting on a fine grid you could start at a coarser and work your way up. This is referred to as Full Multi grid (FMG). One advantage is that the solver start with few degrees of freedom and only moves to a finer grid if needed saving computing time.
2.4.6 Preconditioner

Through this section, methods of solving $Ax = b$ have been explained. The convergence criteria which has been discussed briefly involves eigenvalues. When relaxation methods were explained the criteria was that the largest eigenvalue need to be less than 1. That is the largest eigenvalue for the matrix within the iteration scheme and not the $A$ matrix. Because MG and FMG used relaxation methods the above criteria holds. For the CG methods the spectral condition number defined as $\lambda_{max}/\lambda_{min}$ is used.

The idea with a preconditioner is to lower the maximum eigenvalue and get a better condition number.

$$E^{-1}Ax = E^{-1}b$$

(105)

Solving the above for $x$ is equivalent of solving $Ax = b$. If $E^{-1}A$ has a better condition number than $A$ the problem will converge faster. Important aspects are properties like symmetry and positive definiteness. Even if $A$ has those properties it is not guaranteed that $E^{-1}A$ has them.
3 Method

In this section the method and procedure is explained and described. After a general explanation of the procedure the domain and grid settings are described and finally the solving procedure and testing methods.

The task is to approximate the wind model in a complex urban environment and to check the result and the convergence. The idea is to apply the model on three different domains and test different attributes using CM. The task is divided into three parts.

- In part 1 an empty domain is used to evaluate if the model is solvable without any interference from objects. Different approximations, mesh sizes, elements and solvers according to table (1) is used.

- In part 2 a block is added to the same domain as in step one. Here the results are compared against wind tunnel data from [4]. The settings depend on result from part 1.

- In part 3 the domain is on city scale representing Oklahoma city. Here the focus lie on different solvers.

Throughout the simulation the main idea is to keep everything simple and strive for fast solution times due to the possibility of implementation later on by FOI.

<table>
<thead>
<tr>
<th>Element</th>
<th>Mesh size</th>
<th>DOF $V_0/V_u$</th>
<th>Approximation</th>
<th>$\Delta z/h$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tetrahedral</td>
<td>Coarse</td>
<td>15500/5000</td>
<td>linear</td>
<td>0.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>quadratic</td>
<td>0.5</td>
</tr>
<tr>
<td></td>
<td>Fine</td>
<td>35000/12000</td>
<td>linear</td>
<td>0.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>quadratic</td>
<td>0.4</td>
</tr>
<tr>
<td></td>
<td>Extra fine</td>
<td>310000/103500</td>
<td>linear</td>
<td>0.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>quadratic</td>
<td>0.2</td>
</tr>
<tr>
<td>Brick</td>
<td>Coarse</td>
<td>15700/5000</td>
<td>linear</td>
<td>0.25</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>quadratic</td>
<td>0.5</td>
</tr>
<tr>
<td></td>
<td>Fine</td>
<td>35000/12000</td>
<td>linear</td>
<td>0.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>quadratic</td>
<td>0.35</td>
</tr>
<tr>
<td></td>
<td>Extra fine</td>
<td>310000/103500</td>
<td>linear</td>
<td>0.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>quadratic</td>
<td>0.17</td>
</tr>
</tbody>
</table>
3.1 The Domain

During part 1 and 2 the domain is kept small and imitates a channel flow with closed walls according to figure (6). The size is based on [4] with $0 < x < 0.25[m], 0 < y < 0.075[m]$ and $0 < z < 0.05[m]$. The square block (in part 2) measures $h = 0.025[m]$ and all the domain measurements are divided with $h$, which is the height of the block. The domain become $(10 \times 3 \times 2)$ with the center of the block positioned at $z/h = (2.5, 1.5, 0.5)$.

Figure 6: The normalised domain for part 2. Gray and black indicate closed boundaries.

In part 3 a section of Oklahoma City is used. The Domain contains a variety of complex buildings and spans an area of $700 \times 700 \times 190 [m]$. Figure (7) shows Oklahoma City.

Figure 7: The domain for part 3.
3.1.1 The Grid

In part 1 and 2 different types of elements are tested. The first one is a Tetrahedral element, which is a pyramid with a triangular base. The second element is a brick. CM lacks methods for creating a brick grid for a complex urban environment. Therefore only tetrahedral elements are used in part 3.

The mesh size is varied according to table 1. $\hat{V}_0$ have three times the Degrees of Freedom (DOF) as $\hat{V}_u$ since $\hat{V}_0$ is a vector field $(u, v, w)$ and $\hat{V}_u$ is calculated by solving $\lambda$ which is a scalar field.

![Diagram](image)

Figure 8: Shows $|\hat{V}_{00}|/|\hat{V}_{00,\text{max}}|$ in x-direction without the recirculation zones as a function of $z/h$. The profile is used in part 1 and 2.

The profile seen in figure 8 is the $V_{00}$ field that is used in part 1 and 2. To be able to resolve the flow in the lowest part of the profile, $\Delta z/h$ needs to be < 0.02. This will give a DOF of over $7 \cdot 10^6$ and a computing time well over the criteria for this fast urban model. For the computing time to be practical when the domain is scaled up to city size, $\Delta z/h$ will span between 0.5 and 0.1. This can be justified from studying the expected result in [3]. $\Delta z/h$ is estimated with,

$$\Delta z/h = \frac{Z/h}{n - 1}$$

(106)

were $Z$ is the height of the domain, $h$ is the height of the cube and $n$ is the total number of nodes found when travelling from $z/h = 0$ to $z/h = 2$. Below in figure (9) the two element options are presented. The volumes spanned by the elements are held constant in part 1 and 2.
Figure 9: Shows the computational grid for part 2.
3.2 Solving

Three different types of iterative solvers are tested. First relaxation solvers including both the Jacobi Method (JM) and the Successive Over Relaxation (SOR) with \( w = 0.5, 1.0 \) and 1.5. The second type is the Conjugate Gradi- ent method (CG), which uses search directions to find the solution and finally a multi grid (MG) solver. The procedure starts with JM and more advanced solvers are tested if convergence is not achieved. The solvers will iterate until the relative error is < 0.001.

The computer used has a Intel(R) Xeon(R) W3520 2.67GHz quad core processor with 12 GB memory (RAM) and a 64-bit operating system.

3.2.1 Generating the initial field with recirculation zones

There is a possibility to create a function for each component in \( \hat{V}_0 \) instead of generating the \( \hat{V}_0 \) field with CM. These functions are defined from the equations in section (2.2.2) and needs to take the different zones into consideration. These functions can then be used when \( \hat{V}_u \) is generated.

The numerical procedure is to define different small domains that holds the different recirculation zones and letting CM solve,

\[
\hat{V}_0 = \hat{f}
\]

for each domain. Here \( \hat{V}_0 = (u_0, v_0, w_0) \) and \( \hat{f} \) containing functions for each velocity component and is defined for each domain according to section (2.2.2). This give either a linear or quadratic approximation of the wind field between the same grid points that later are used for the Poisson equation.

Instead of using small domains if statements can be used to define \( \hat{f} \) depending on position. Then equation (107) is solved only for one domain. This is used for part 1 and 2.

Because \( \hat{V}_0 \) was derived from an empirical study within an urban environment, \( V_{00} \) used in the equations for \( \hat{f} \) is the mean flow from [4] seen in figure 8. In part 1 there is no buildings so \( \hat{V}_0 = V_{00} \).

In part 3 \( \hat{V}_0 \) is directly imported from results using algorithm already developed by FOI, so there is no need to solve equation (107). The \( \hat{V}_0 \) field is interpolated into the grid. The reason is that CM can not generate the recirculation zones automatic because the program need to be able to identify rectangular shapes from buildings within the urban environment and make decisions regarding the parameters governing the recirculation zones. In CM, you are required to define each zone individually.

The boundary condition used when solving the \( \hat{V}_0 \) field in part 1 and 2 is \( \hat{V}_0 = \hat{f} \) for the complete boundary.
3.2.2 Solving the Poisson equation

Equation (108) is solved either with a linear or quadratic approximation.

\[
\frac{1}{2\alpha^2} \frac{\partial^2 \lambda}{\partial x^2} + \frac{1}{2\beta^2} \frac{\partial^2 \lambda}{\partial y^2} + \frac{1}{2\gamma^2} \frac{\partial^2 \lambda}{\partial z^2} = -\left( \frac{\partial u_0}{\partial x} + \frac{\partial v_0}{\partial y} + \frac{\partial w_0}{\partial z} \right) \quad (108)
\]

The derivatives on the R.H side is numerically calculated by CM from the $\hat{V}_0$ field previously solved or imported. $\nabla \lambda$ is then numerically calculated by CM and the Euler equations are updated according to equation (109) to give $\hat{V}_u$ which is mass conserved.

\[
\hat{V}_u = \hat{V}_0 + \hat{b} \nabla \lambda \quad (109)
\]

Where $\hat{b} = (1/2\alpha^2, 1/2\beta^2, 1/2\gamma^2)$. The boundary conditions are set according to section (2.2.4) with $\hat{n} \cdot \hat{c} \nabla \lambda = - (u_0 + v_0 + w_0) \cdot \hat{n}$ for the wall and $\lambda = 0$ for the free boundary. The constants $\alpha, \beta$ and $\gamma$ controls the mass redistribution. $\alpha$ and $\beta$ are always equal because these two controls the flow in the horizontal direction. By changing the ratio towards $\gamma$ you can control how the $\hat{V}_0$ field is altered. If $\gamma < \alpha, \beta$ the flow will mainly go over the cube instead of around the side. So these settings control the stability of the air mass. For this test the assumption is that $\alpha = \beta = \gamma = 1$, which means the mass is equally distributed around the cube. A detailed description about stability is found in [1].
3.3 Verification and testing

To be able to compare the numerical error in part 1,

\[ W = \frac{\sum (\hat{V}_{00}(z_i) - \hat{V}_u(z_i))}{\hat{V}_{00}(z_i)} \]

is used as a relative error parameter. Here \( \hat{V}_{00} \) is the mean wind profile from [4] and \( n \) is the total number of extracted measurements.

In part 2 the field is compared against reference data. The measurement points are presented in figure 10. They are positioned so reference data can be used from [4]. Not all measured points will be presented in the result.

Figure 10: Top view of measurement points on the upwind and downwind side.

Main features of the flow is described in [4]. Features like a vortex upstream within the shear flow (point A in figure 11), side (point C in figure 11) and roof (point D in figure 11) vortex, the Arch vortex (point B in figure 11) directly downstream and the Horseshoe vortex (point E in figure 11) beginning at the sides and stretches downstream. Beside recognising the main flow features \( \hat{V}_u \) will be plotted against \( \hat{V}_{00} \) to see influence by buildings and against experimental data from [4] to test the accuracy.

Figure 11: Illustrates the main features of a turbulent flow around a cube.
4 Result

This section is divided into the three subsections. First the result from part 1 in presented, that handled an empty domain. Secondly the result from part 2 were a block and circulation zones were used and finally, the result from part 3 that tested the model in an urban environment.

4.1 Empty domain, part 1

Table 2 below presents the result from part 1. The estimated error $W$ decreased when DOF increased. Even if the approximation was changed from linear to quadratic for the same DOF the error is not getting any smaller. The same test was performed for a laminar wind profile (Appendix E) which has a natural scale higher than the estimated $\hat{V}_u$. In that case the estimated error $W$ was below $10^{-4}$ for the quadratic and between $10^{-2}$ and $10^{-4}$ for the linear approximation.

Table 2: Computing time and error estimate $W$ for the initial field in figure 8. $\text{inf}$ means the relative error converged to infinity i.e the algorithm did not converge. $\text{inf}/4$ means the $V_0$ field was solved with SOR and the $V_u$ field was solved with JM. A single number means the total time.

<table>
<thead>
<tr>
<th>Element</th>
<th>Mesh</th>
<th>Approximation</th>
<th>Computer time</th>
<th>$W$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>$V_0/V_u$ [s]</td>
<td>$V_0 + V_u$ [s]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Jacobi</td>
<td>SOR (w=1)</td>
</tr>
<tr>
<td>Tetrahedral</td>
<td>Coarse</td>
<td>Linear</td>
<td>inf/4</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Quadratic</td>
<td>inf/inf</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>Fine</td>
<td>Linear</td>
<td>inf/5</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Quadratic</td>
<td>inf/inf</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>Extra fine</td>
<td>Linear</td>
<td>inf/18</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Quadratic</td>
<td>inf/inf</td>
<td>16</td>
</tr>
<tr>
<td>Brick</td>
<td>Coarse</td>
<td>Linear</td>
<td>inf/4</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Quadratic</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>Fine</td>
<td>Linear</td>
<td>inf/4</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Quadratic</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>Extra fine</td>
<td>Linear</td>
<td>inf/18</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Quadratic</td>
<td>34</td>
<td>18</td>
</tr>
</tbody>
</table>

When studying the Extra fine setting, the result suggests that the computing time is higher when a quadratic approximation is used instead of a linear even if the same DOF is used. The SOR seems to be a more reliable solver compared to JM. The JM has problems with the initial field when using tetrahedral elements and the linear approximation for the brick elements.
4.2 Block domain, part 2

From [3] the natural scale for $V_u$ is estimated to be $\Delta z/h \approx 0.2$. Therefore the settings from table 1 are used.

Table 3: Computing time for the SOR and MG solver.

<table>
<thead>
<tr>
<th>Element</th>
<th>Mesh size</th>
<th>approximation</th>
<th>Computer time</th>
<th>$V_0 + V_u [s]$</th>
<th>$V_0 + V_u [s]$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>SOR (w=1)</td>
<td></td>
<td>MG (SOR, w=1)</td>
</tr>
<tr>
<td>Tetrahedral</td>
<td>Coarse</td>
<td>linear</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Fine</td>
<td>linear</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Extra fine</td>
<td>linear</td>
<td>23</td>
<td>21</td>
<td></td>
</tr>
<tr>
<td>Brick</td>
<td>Coarse</td>
<td>linear</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Fine</td>
<td>linear</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Extra fine</td>
<td>linear</td>
<td>16</td>
<td>18</td>
<td></td>
</tr>
</tbody>
</table>

In table 3 the SOR and MG solver managed to solve the problem with circulation zones without any problems. The computer time is similar to table 2 with an increase when the DOF is increased.
Figure 12: $\hat{V}_0/\hat{V}_{00,\text{max}}$ as a function of $x, z$ at $y = 1.5$.

Figure 13: $\hat{V}_0/\hat{V}_{00,\text{max}}$ as a function of $x, y$ at $z = 0.5$.

The Displacement zone from figure 3 is visible in figure 12 and 13. It reaches a height of $0.6H$ on the windward side of the cube with $V_0 = 0$. On the leeward side the velocity within the Cavity zone is $-u_{00}(H)$ at $x/h = 3$ and decays to zero when reaching the downwind boundary. The velocity on the upstream boundary of the Wake zone is zero and increases when travelling downstream until it hits the downwind boundary where it increases fast to the ambient velocity. The figures are consistent with the equations described in section (2.2.2).
Figure 14: $\hat{V}_u/\hat{V}_{00,\text{max}}$ as a function of $x, z$ at $y = 1.5$.

Figure 15: $\hat{V}_u/\hat{V}_{00,\text{max}}$ as a function of $x, y$ at $z = 0.5$.

Figure 14 and 15 show the mass conserving field. The wind flows over and around the cube. On the wind side, there is a tendency for a weak back flow that can be confirmed with a similar simulation in [7]. On the leeward side the entire Wake zone has become a back flow and the Arch vortex in figure 11 is clearly visible in the region behind the cube. There are no visible vortexes on the sides or on top.
Figure 16: $u/u_{0,\text{max}}$ as a function of $z/h$ for tetrahedral elements half a cube length upstream from the leading edge in the vertical center plane of the channel.

Figure 17: $u/u_{0,\text{max}}$ as a function of $z/h$ for brick elements half a cube length upstream from the leading edge in the vertical center plane of the channel.

The velocity component in x-direction is plotted in figures 16 and 17. Basically all change in velocity for the initial field $u_{00}$ is below the top of the cube ($z/h < 1$). There is a big difference between the estimated mass conserving wind field and the experimental profile. The vortex within the shear flow, situated at $z/h \approx 0.45$ is positioned too high and the magnitude is too low compared to the experimental profile. There is a tendency for a decrease in velocity above
Daniel Eriksson

the vortex compared to an increase for the experimental data.

In figure 17 the solver has problem resolving the vortex at $z/h = 0.45$ with a coarse mesh for brick elements. The nodes are missing the position with the strongest back flow. For both brick and tetrahedral elements the nodes are visible when the velocity change is rapid.

Figure 18: $u/u_{00,\text{max}}$ as a function of $z/h$ a quarter of a cube length upstream from the leading edge of the cube in the vertical center plane of the channel.

Figure 19: $u/u_{00,\text{max}}$ as a function of $z/h$ a quarter of a cube length upstream from the leading edge of the cube in the vertical center plane of the channel.
Figures 18 and 19 show the velocity profile a quarter of a cube length upstream from the leading edge of the cube in the vertical center plane of the channel. The results are similar to figures 16 and 17 with the vortex position to high and visible nodes for the Coarse setting. The magnitude of the flow is still less than the experimental profile. The brick elements seem to handle the $u/u_{00,\text{max}}$ profile better than tetrahedral for Course and Fine mesh setting.
Figure 20: $u/u_{00,\text{max}}$ as a function of $z/h$ half a cube length downstream from the trailing edge of the cube.

Half a cube length downstream from the trailing edge of the cube the wind profile passes three zones in figures 20 and 21. There is a back flow in the Cavity zone positioned in the region $0 < z/h < 0.80$. The Arch vortex is positioned in the center of the Wake zone positioned in the region $0.80 < z/h < 0.90$. The magnitude is poorly estimated both below and above the vortex. It is clear that with a finer grid the numerical error is reduced within the Cavity zone. For the
brick element, the magnitude of $V_u$ differs substantially compared to Fine and Extra fine.

In figure 22 and 23 the numerical error is reduced. The Arch vortex is still present creating a back flow for $\Delta z/h < 0.4$. Compared to the experimental profile the height should have been reduced significantly. There is a small visible difference between the brick and the tetrahedral mesh in the wake zone.
4.3 City, part 3

Only $\hat{V}_u$ is solved because $\hat{V}_0$ is imported. The JM solver is therefore included. Here the grid was generated with CM own grid generator.

Table 4: The computer time and number of iterations for different solvers for DOF=782 000 and DOF=2 100 000.

<table>
<thead>
<tr>
<th>Solver</th>
<th>DOF=782 000</th>
<th>$\hat{V}_u$</th>
<th>$\hat{V}_0$</th>
<th>Total time [min:s]</th>
<th>Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jacobi</td>
<td></td>
<td>$\hat{V}_u$</td>
<td>$\hat{V}_0$</td>
<td>-</td>
<td>&gt;10000</td>
</tr>
<tr>
<td>SOR (w=0.5)</td>
<td></td>
<td></td>
<td></td>
<td>-</td>
<td>&gt;10000</td>
</tr>
<tr>
<td>SOR (w=1.0)</td>
<td></td>
<td></td>
<td></td>
<td>6:30</td>
<td>8997</td>
</tr>
<tr>
<td>SOR (w=1.5)</td>
<td></td>
<td></td>
<td></td>
<td>3:10</td>
<td>3067</td>
</tr>
<tr>
<td>CG</td>
<td></td>
<td></td>
<td></td>
<td>1:51</td>
<td>96</td>
</tr>
<tr>
<td>MG (One &quot;Two grid cycle&quot;)</td>
<td></td>
<td></td>
<td></td>
<td>2:11</td>
<td>7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Solver</th>
<th>DOF=2 100 000</th>
<th>$\hat{V}_u$</th>
<th>$\hat{V}_0$</th>
<th>Total time [min:s]</th>
<th>Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>CG</td>
<td></td>
<td>$\hat{V}_u$</td>
<td>$\hat{V}_0$</td>
<td>3:00</td>
<td>463</td>
</tr>
<tr>
<td>MG (One &quot;Two grid cycle&quot;)</td>
<td></td>
<td></td>
<td></td>
<td>3:43</td>
<td>7</td>
</tr>
</tbody>
</table>

Every solver performed well except for the JM and SOR (w=0.5) which converged very slowly exceeding the iteration limit of 10 000. The CG and MG performed best with a solution time of around 2 min with a DOF of 782 000. The MG solver had a longer preparation time than CG and used less iterations. The effectiveness of MG can be observed by comparing the number of iterations between the different mesh settings. The CG is slightly faster than MG.

Figure 24: $\hat{V}_u$ in Oklahoma City

In figure 24 the $\hat{V}_u$ field can be observed circling the two tallest buildings in a part of Oklahoma City.
Figure 25: A close-up picture showing $\hat{V}_u$ circling a building in Oklahoma City.

In figure 25 a backflow is visible which is part of the lower part of the Arch Vortex. Both the Cavity and Wake Zone is visible.

Figure 26: $\hat{V}_u$ in Oklahoma City

In figure 26 the flow is following the streets and adapting to the surrounding buildings. The flow is more complex compared to part 2. Here there is a combination of multiple Displacement, Cavity and Wake zones. The flow is also coming in at an angle further complicating the rendering of the initial field. Focusing on the two buildings in the center of figure 26 the Displacement zone upstream is visible with an area with less velocity. Also downstream there are areas effected by recirculation zones.
5 Discussion

This model uses only the mass conservation law and approximates the energy and momentum law with an empirical diagnostic study of the wind fields within urban cities. If a decrease in accuracy is accepted this is a good way of decreasing the computing time to minutes instead of days when compared to a urban LES simulation of Stockholm found in [4].

There are differences in performance between the relaxation methods, CG and the MG method used in this work. The JM was not able to solve the initial field. This suggest that $E_{jac}$ contains an eigenvalue > 1. This could be a problem if the JM is used within the MG solver. Comparing computing times the MG and CG was faster than JM and SOR and should be the natural choice. The effectiveness of MG can be seen when comparing the number of iteration used. Even if the DOF was increased, the same number of iterations was used.

In part 1 $\hat{V}_0$ has a natural scale of $\Delta z/h \approx 0.02$. When solving $\hat{V}_u$ with $\Delta z/h > 0.02$ the error was not reduced when changing the approximation from linear to quadratic for the same DOF. In section (4.1) a laminar wind profile was used with a natural scale of $\Delta z/h \approx 0.5$. Here the error decreased when changing from linear to a quadratic approximation for the same DOF. Still the natural scale for this model when studying figures 14 and 15 is near $0.1[z/h]$, so $\Delta z/h$ needs to be less than 0.1 to achieve a decrease in error for a quadratic approximation. This means that there is no need for a quadratic approximation because $\Delta z/h < 0.1$ will give an impractical total solution time. Also a linear approximation is much easier to use since $\nabla \phi$ is constant, as shown in the theory section.

From figures 16 to 23 there is nothing that suggest that tetrahedral elements is better than brick elements. Both have areas were they perform poorly. What can be seen is that a smaller $\Delta z/h$ give more reliable results with less numerical overshoot. The dilemma is that to be able to solve Oklahoma City in 3 minutes with 2 100 000 DOF, the nodes need to be concentrated around buildings with $\Delta z/h \approx 0.1$ to generate a good approximation. Otherwise with a equally spaced grid, $\Delta z/h \approx 0.1$ and a height of 10 [m] for the smallest building in the Oklahoma domain gives a DOF of 93 000 000. This give an impractical solution time way over 3 minutes. A larger $\Delta z/h$ could be used if the dispersion model can handle results that are less accurate.

When compared to wind tunnel tests the result reveal the importance of a good approximation of the initial field because $\hat{V}_u$ follows $\hat{V}_0$ when not influenced by any boundary condition. The model fail too anticipate the roof and side vortex but finds the Arch and upstream vortex. The upstream vortex is positioned too high, giving a large back flow area with underestimated velocities. The Arch vortex have a god position but the back flow generated is too weak and extends too far downstream. Overall the model have difficulties approximating the correct velocity. This evaluation was against wind tunnel measurement over an aerodynamically smooth cube and the model was created by an empirical diagnostic study from a complex urban environment. So the model may perform better when compared to measurement taken from a real urban city.
6 Conclusion

Generating wind fields in an urban environment is very complicated. Turbulence is generated by objects stretching from large buildings down to the smallest leaf. This work has shown that there is a way to solve the mass conservative wind field model for a large urban environment using Comsol Multiphysics and the Finite Element Method in a reasonably short time and be able to resolve and visualise the large features in the flow generated by buildings.

The CG and MG (SOR) performed best. Important here is to use the SOR instead of the JM when using MG, the solution will otherwise diverge to infinity. If the grid size is larger than the natural scale there is no need for a quadratic approximation to achieve more accuracy with a similar solution time.

To achieve a reasonable short computing time there are a few things to consider. First the grid size need to be considered carefully. A larger $\Delta z/h$ than 0.1 will give less accuracy and more numerical overshoot but a short solution time. A wind field could still give sufficient desperation results even if poorly resolved vortexes are used. To be able to optimize $\Delta x/h$, $\Delta y/h$ and $\Delta z/h$ for a short solution time, dispersion testing should be performed.

Secondly if the complete domain is used, it is clear that a varying grid size is needed to keep a low number of calculation points, to achieve a short solution time and still be able to resolve the main features accurately. The height of the domain could be made smaller because some distance above the buildings $\hat{V}_u = \hat{V}_0$. So their is no need to solve the domain above that height. This will reduce the number of calculation points needed and further decreasing the computing time. This could mean that a varying grid is not longer needed. Still more investigation is needed to find the height where $\hat{V}_u = \hat{V}_0$ due to different stability conditions in the atmosphere over time.

Before any alterations are made to the model to increase the accuracy, the result should be evaluated against a real urban wind field and a real dispersion test. If the accuracy is still not sufficient a modified version is evaluated in [7] with better agreement to wind tunnel tests over a cube.

If the computing time and precision is sufficient for dispersion modelling, after testing against an urban environment this model could be a powerful tool for fast emergency response simulations within ARGOS.

6.1 Future work

- Evaluation against an urban wind field and real dispersion.
- Alter the equations governing the recirculation zones according to [7] and evaluate against urban wind field and real dispersion.
- Investigate why the result differs from experimental wind tunnel data.
A Appendix

A.1 Global minimum

Show that:

\[ f(x) = \frac{1}{2} x^T Ax - b^T x \]  

has a global minimum so \( x \) minimizes \( f(x) \) for all \( \epsilon \).

\[
\begin{align*}
    f(x + \epsilon) &= \frac{1}{2} (x + \epsilon)^T A (x + \epsilon) - b^T (x + \epsilon) \\
    &= \frac{1}{2} x^T A x + \frac{1}{2} x^T A \epsilon + \frac{1}{2} \epsilon^T A x + \frac{1}{2} \epsilon^T A \epsilon - b^T x - b^T \epsilon \\
    &= \frac{1}{2} x^T A x + \frac{1}{2} \epsilon^T A \epsilon - b^T \epsilon \\
    &\quad + (\frac{1}{2} x^T A x - b^T x) + (\epsilon^T A x + \frac{1}{2} \epsilon^T A \epsilon - b^T \epsilon) \\
    &= f(x) + \frac{1}{2} \epsilon^T A \epsilon
\end{align*}
\]

The second term on the right hand side is always positive because \( A \) is positive definite. That means for all \( \epsilon \neq 0 \), \( f(x + \epsilon) > f(x) \). So equation 111 has a global minimum.
A.2 Solution in n steps

With the assumption that $d_i^T A d_j = 0$ for all $i \neq j$. Show that it takes at most $n$ search directions to arrive at the solution.

The error $e_0$ can be expressed as a linear combination,

$$ e_0 = \sum_{j=0}^{n-1} \delta_j d_j $$

(113)

multiplying the expression with $d_k^T A$,

$$ d_k^T A e_0 = \sum_{j=0}^{n-1} d_k^T A \delta_j d_j $$

(114)

$$ \delta_k = \frac{d_k^T A e_0}{d_k^T A d_k} \quad d_k^T A = \frac{d_k^T A (e_0 + \sum_{j=0}^{n-1} \delta_j d_j)}{d_k^T A d_k} $$

(115)

Comparing the result with equation (86) shows that $\alpha_i = -\delta_i$. That means each time a new $x$ value is derived the error gets smaller. Let’s write the error for $x_i$,

$$ e_i = \sum_{j=0}^{n-1} \delta_j d_j + \sum_{j=0}^{i-1} \alpha_j d_j $$

(116)

Here the first term is the error at the starting position $x_0$ expressed by summing all the search vectors from $x_0$ to $x$. $\alpha_i$ is then substituted for $-\delta_i$.

$$ e_i = \sum_{j=0}^{n-1} \delta_j d_j - \sum_{j=0}^{i-1} \delta_j d_j $$

(117)

If we instead look at $i = n$ the error at $x_n$ become zero. That means the CG method converges in n iterations with the requirement that $d_i^T A d_j = 0$ for all $i \neq j$. 

42
A.3 Gram-Schmidt Conjugation

When constructing $A$-orthogonal search vectors we first define a set of $n$ vectors $(v_0, v_1, ..., v_{n-1})$. Set the first search direction $d_0 = v_0$ and define,

$$d_i = v_i + \sum_{k=0}^{i-1} \beta_{ik} d_k$$

(118)

Were $d_k$ are the previous search directions and $\beta_{ik}$ are the correction making $d_i$ $A$-orthogonal to the previous search directions. To find $\beta$ the expression is multiplied from the right with $Ad_j$.

$$d_i^T Ad_j = v_i^T Ad_j + \sum_{k=0}^{i-1} \beta_{ik} d_k^T Ad_j$$

(119)

The search direction $d_i$ is only compared to previous $(k < i)$ which means $d_i^T Ad_j = 0$, the expression becomes,

$$0 = v_i^T Ad_j + \beta_{ij} d_j^T Ad_j \quad (j < i)$$

(120)

Which gives,

$$\beta_{ij} = -\frac{v_i^T Ad_j}{d_j^T Ad_j} \quad (j < i)$$

(121)

Where $j < i$. The vectors used to construct the search directions are the residuals when using CG, so $v_i = r_i$. This means equation (121) can be written as,

$$\beta_{ij} = -\frac{r_i^T Ad_j}{d_j^T Ad_j} \quad (j < i)$$

(122)
A.4 Only the previously residual

In this section equation (88) is rewritten and it is shown that only the previous residual is needed to calculate $\beta_{ij}$. From equation (83) the residual can be expressed as

$$r_{j+1} = -Ae_{j+1} = -A(e_j + \alpha_j d_j) = r_j - \alpha_j Ad_j.$$  \hspace{1cm} (123)

Multiplying the expression with $r_i^T$ from the left gives,

$$r_i^T r_{j+1} = r_i^T r_j - r_i^T \alpha_j Ad_j.$$  \hspace{1cm} (124)

Rearranging,

$$r_i^T Ad_j = (r_i^T r_j - r_i^T r_{j+1}) \frac{1}{\alpha_j} \hspace{1cm} (125)$$

We get,

$$r_i^T Ad_j = r_i^T r_j - r_i^T r_{j+1} \hspace{1cm} (j = i)$$

$$r_i^T Ad_j = -r_i^T r_{i-1} \frac{1}{\alpha_{i-1}} \hspace{1cm} (j = i-1)$$

$$r_i^T Ad_j = 0 \hspace{1cm} (j > i, j < i-1) \hspace{1cm} (126)$$

Equation (126) and (122) for $j < i$ give,

$$\beta_{i,j} = \frac{r_i^T r_j}{\alpha_{i-1}^2} \hspace{1cm} (j = i-1)$$

$$\beta_{i,j} = 0 \hspace{1cm} (j < i-1) \hspace{1cm} (127)$$

The expression for $\beta_{ij}$ can be rewritten by inserting the expression for $\alpha$ from equation (86),

$$\beta_i = \frac{r_i^T r_i}{\alpha_{i-1}^2} \hspace{1cm} (128)$$

We can simplify this expression even more by multiply equation (117) with $-d_i^T A$.

$$-d_i^T Ae_j = -d_i^T A \sum_{j=1}^{n-1} \delta_j d_j$$

$$d_i^T r_j = -\sum_{j=1}^{n-1} \delta_j d_i^T Ad_j$$

$$d_i^T r_j = 0 \hspace{1cm} (i < j) \hspace{1cm} (129)$$

This means that the search direction $d_i$ is orthogonal with all previous residuals. Use equation (118) with $v_i = r_i$ and taking the inner product with $r_i$ gives,

$$d_i^T r_i = r_i^T r_i + \sum_{k=0}^{i-1} \beta_{i,k} d_k^T r_i \hspace{1cm} (130)$$

This means that equation (128) can be written as,

$$\beta_i = \frac{r_i^T r_i}{\alpha_{i}^2} \hspace{1cm} \frac{1}{\alpha_{i}} \hspace{1cm} (131)$$

This means that only the previous residual is needed.
A.5 The computing time and error estimate for a laminar initial wind profile

Table 5: Shows Computing time and error estimate $W$ for a laminar initial wind profile for part 1

<table>
<thead>
<tr>
<th>Element</th>
<th>Mesh</th>
<th>approximation</th>
<th>computer time $V_0/V_u$</th>
<th>$W$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>$V_0/V_u$ &amp; $V_u$</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Jacobi</td>
<td>SOR (w=1)</td>
</tr>
<tr>
<td>Tetrahedral</td>
<td>Coarse</td>
<td>linear</td>
<td>inf/2</td>
<td>2/1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>quadratic</td>
<td>inf/inf</td>
<td>1/1</td>
</tr>
<tr>
<td></td>
<td>Fine</td>
<td>linear</td>
<td>inf/2</td>
<td>2/2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>quadratic</td>
<td>inf/inf</td>
<td>2/2</td>
</tr>
<tr>
<td></td>
<td>Extra fine</td>
<td>linear</td>
<td>inf/3</td>
<td>4/3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>quadratic</td>
<td>inf/inf</td>
<td>2/3</td>
</tr>
<tr>
<td>Brick</td>
<td>Coarse</td>
<td>linear</td>
<td>inf/1</td>
<td>1/1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>quadratic</td>
<td>inf/1</td>
<td>1/1</td>
</tr>
<tr>
<td></td>
<td>Fine</td>
<td>linear</td>
<td>inf/2</td>
<td>2/1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>quadratic</td>
<td>3/2</td>
<td>2/2</td>
</tr>
<tr>
<td></td>
<td>Extra fine</td>
<td>linear</td>
<td>inf/2</td>
<td>3/2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>quadratic</td>
<td>3/3</td>
<td>3/2</td>
</tr>
</tbody>
</table>
References


