3D Modeling of Urban Areas for Built Environment CFD Applications using COMSOL

A.W.M. (Jos) van Schijndel
Eindhoven University of Technology
P.O. Box 513; 5600 MB Eindhoven; Netherlands, A.W.M.v.Schijndel@tue.nl

Abstract:
This paper presents the modeling and simulation results of wind velocities in urban areas. The main idea was to build random generated urban areas for studying the influence of different urban geometries, from relative open to more dense, on wind profiles. The Spalert-Allmaras (spf) turbulent flow model turned out to be suitable for domains up to 50 x 50 x 100 m^3. The k-ε turbulence model provided reliable results for domains up to 100 x 600 x 2000 m^3. We conclude that the presented methodology is promising for 3D CFD modeling of urban areas for built environment applications.

Keywords: Urban area, wind, CFD, buildings

1. Introduction

Building physics aims to study the built environment on several scales: mm-scale: material physics; the m-scale: indoor climate and energy physics; the km-scale: urban physics. Figure 1 provides an overview of these scales.

Figure 1. Relevant scales in building physics

Looking at current use of Comsol, most building physics applications are related with the material physics scale, some applications at the m-scale, and only very few applications at the km scale. For the latter we refer for example to the Geomechanics and the Subsurface Flow Modules that are designed for geophysical and environmental phenomena studies.

In this paper we present a methodology to build 3D urban area models based on random placed buildings for Built Environment Applications. The methodology was: Step 1, creating 3D urban area geometries using Comsol with a MatLab script. Several types of urban areas ranging from low to high building densities can be build. A typical scale is of order ~1 km^3. Step 2, importing the geometry in Comsol and automatic mesh generation of these geometry models. Step 3, adding boundary values and simulation of the wind profiles.

The paper has the following outline: Section 2 shows CFD modeling considerations; Section 3 provides the details on how to generate random urban areas. Section 4 presents the results of two applications using Spalert-Allmaras (spf) and k-ε turbulence models respectively.

2. CFD modeling considerations

Every scale level has specific model limitations. The k-ε turbulence model relies on several assumptions, the most important of which are that the Reynolds number is high enough and that the turbulence is in equilibrium in boundary layers, which means that production equal dissipation. These assumptions limit the accuracy of the model because they are not always true. It does not, for example, respond correctly to flows with adverse pressure gradients which can result in under predicting the spatial extension of recirculation zones (Wilcox 1998). Furthermore, in the description of rotating flows, the model often shows poor agreement with experimental data (Driver et al. 1985). In most cases, the limited accuracy is a fair trade-off for the amount of computational resources saved compared to more complicated turbulence models.

2.1 Wall Functions (Comsol 2012)

The flow close to a solid wall is for a turbulent flow is very different compared to the free stream. This means that the assumptions used to derive the k- ε model are not valid close to walls. While it is possible to modify the k- ε model so that it describes the flow in wall regions (see The Low Reynolds Number k- ε Turbulence Model), this is not always desirable because of the very high resolution requirements that follows. Instead, analytical expressions are used to describe the flow at the walls. These expressions are known as wall functions.

The wall functions in COMSOL are such that the computational domain is assumed to start a distance δ_w from the wall (see Figure 2).

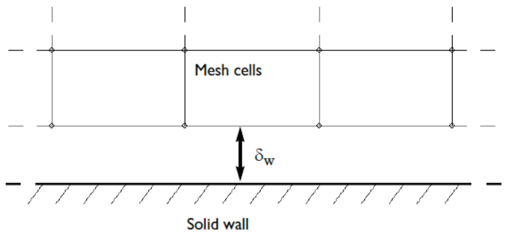


Figure 2. Distance from wall to computational domain

The distance δ_w is automatically computed so that $\delta_w^+ = \rho u_\tau \delta_w/\mu$, where $u_\tau = C_\mu^{1/4}\sqrt{k}$ is the friction velocity, becomes 11.06. This corresponds to the distance from the wall where the logarithmic layer meets the viscous sublayer (or rather would meet if there was not a buffer layer in between). δ_w is limited from below so that it never becomes smaller than half of the height of the boundary mesh cell. This means that u_τ can become higher than 11.06 if the mesh is relatively coarse. We refer to figure 6 where the wall resolution is shown. Please note that the minimum values correspond with 11.06. An experienced user can directly evaluate the quality of the solution. A more inexperienced user can try to improve the results by lowering the wall lifts to 11.06.

2.2 Wind profile at inlet

The following standard wind profile at the inlet was used (see table 1):

Table 1 Wind profile at inlet

Description	Value
Normal inflow velocity	(0.912/0.41)*log((z + 0.1)/0.1)

2.3 Verification and validation

The simulation results may be verified by two important checks: (1) Investigate the solution to check that δ_w is small compared to the dimension of the geometry. (2) Also check that is 11.06 on most of the walls. If δ_w^+ is much higher over a significant part of the walls, the accuracy might become compromised. Both the wall lift-off, δ_w, and the wall lift-off in viscous units, δ_w^+, are available as results and analysis variables. For further information see Kuzmin et al. (2007) and Grotjans et al. (1998).

The validation of CFD models at the urban scale is still problematic due to the limited possibilities to measure and visualize actual 3D wind patterns in real urban areas. For further information see Briggen et al. (2009).

3 Building the urban area geometries

The idea was to build random generated urban areas for studying the influence of different urban geometries, from open areas to more dense ones. MatLab was used to create a Comsol file containing the geometry of the urban area. The MatLab code is far from trivial and is therefore provided in the Appendix. The main parameters are: nB: number of buildings; xp, yp x,y positions of a building; sx, sy, hz: width, length and height of a building. Figure 3 presents an exemplarily output of the mentioned code of the appendix.

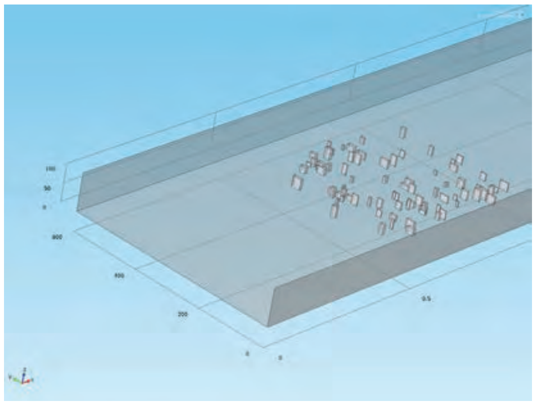


Figure 3. Result of a random urban area.

4 Applications

We started with small urban areas and tested both Spalert-Allmaras (spf) and k-ε turbulence models. If both models provided satisfactory results we increased the scale of the domain.

4.1 Spalert-Allmaras

The largest domain that still provided satisfactory results in case of the spf model, is shown figure 4.

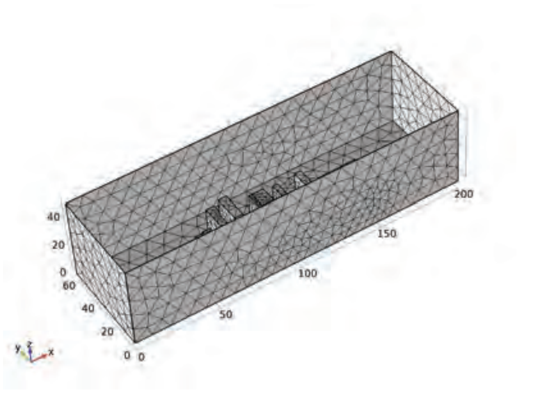


Figure 4. The mesh

After simulation it is possible to check the quality of the solution of the spf model. The dimensionless distance to the cell center is provided in figure 5.

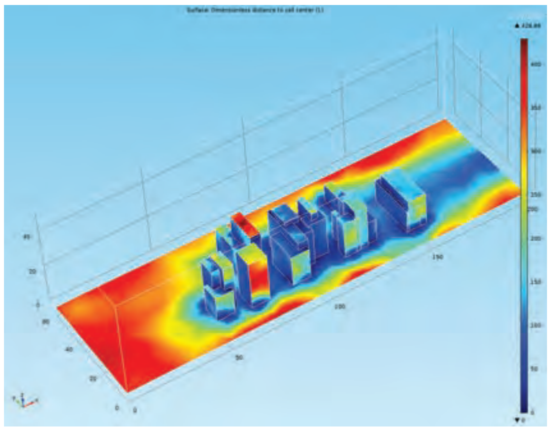
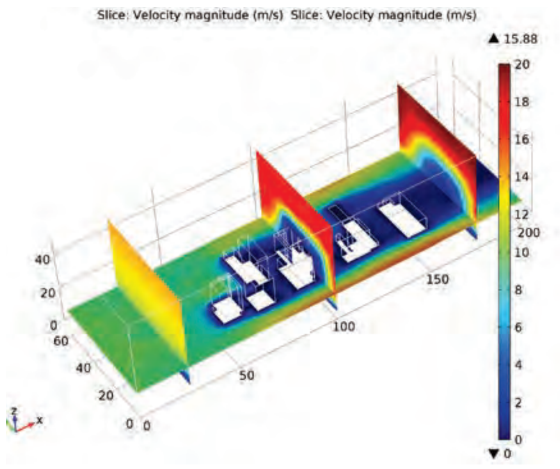


Figure 5. The dimensionless distance to the cell center

It is observed that building facades perpendicular to the prevailing wind direction have the most accurate result. Some facades parallel to the prevailing wind direction and higher wind speeds nearby have less accuracy.

Figure 6 and 7 show respectively the velocity magnitude and the corresponding streamlines.

After increasing the scale of the domain we got problems with obtaining accurate results using the spf turbulence model. Further research is necessary to investigate the limitations of this model. We proceed with the k- ε turbulence model. This model seems to be better suitable for even larger domains. This is shown in the next Section.


Slice: Velocity magnitude (m/s) Slice: Velocity magnitude (m/s)
▲ 15.88
▼ 0
Figure 6. The velocity magnitude

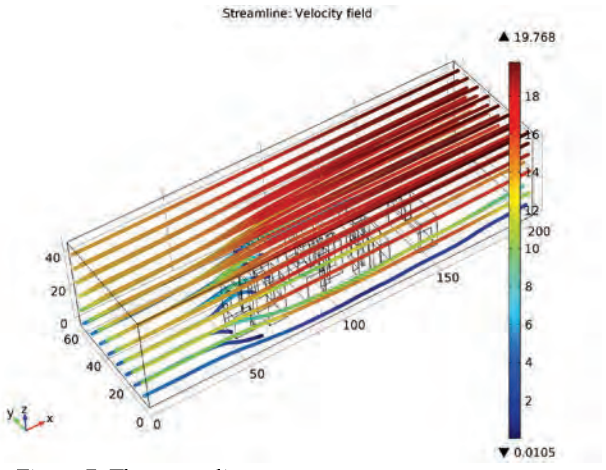


Figure 7. The streamlines.

4.2 k-ε turbulence

After increasing the scale of the domain, figure 3 shows the buildings and the domain.

Figure 8 shows the wall lift-off in viscous units. Figure 9 provides the velocity magnitude. Furthermore the appendix contains 4 results of the velocity magnitude, turbulent kinetic energy and turbulent dissipation rate.

From figure 8, it is observed that only the building facades in front of the domain and perpendicular to the prevailing wind direction have the most accurate results. The rest seems to have less accurate results. Refining the grid may improve the accuracy. The best way to validate the model are experiments in situ. As mentioned in Section 2.3 this is still problematic. Another simpler but less robust option could be benchmarking the model using several other CFD software packages.

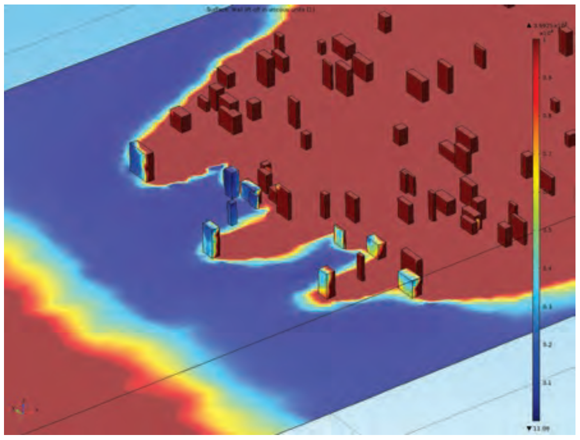


Figure 8. The wall lift-off in viscous units, δ_w^+

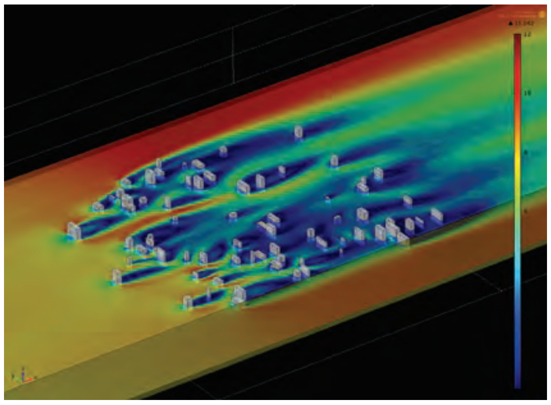


Figure 9. The velocity magnitude

5 Conclusion

This paper presents the modeling and simulation results of wind velocities in urban areas. The main idea was to build random generated urban areas for studying the influence of different urban geometries, from relative open to more dense, on wind profiles. The Spalert-Allmaras (spf) turbulent flow model turned out to be suitable for domains up to 50 x 50 x 100 m^3. The k-ε turbulence model provided reliable results for domains up to 100 x 600 x 2000 m^3.

We conclude that the presented methodology is promising for 3D CFD modeling of urban areas for built environment applications.

Acknowledgement

Ruud Börger of Comsol Netherlands is greatly acknowledged for his support with the CFD modeling.

References

Briggen, P.M., Blocken, B.J.E. & Schellen, H.L. (2009). Wind-driven rain on the facade of a monumental tower: numerical simulation, full-scale validation and sensitivity analysis. Building and Environment, 44(8), 1675-1690.

Comsol (2012), CFD Module User's Guide, www.comsol.com

CFD-online (2012) http://www.cfd-online.com/Wiki/Main_Page

Driver, D.M. and H.L. Seegmiller, (1985) Features of a Reattaching Turbulent Shear Layer in Diverging Channel Flow," AIAA Journal, vol. 23, pp. 163–171.

Grotjans, H. and F.R. Menter, (1998). Wall Functions for General Application CFD

Codes," ECCOMAS 98, Proceedings of the Fourth European Computational Fluid

Dynamics Conference, John Wiley & Sons, pp. 1112–1117, 1998.

HAMLab (2012), http://archbps1.campus.tue.nl/bpswiki/index.php/Hamlab

Kuzmin, D., O. Mierka, and S. Turek, (2007). On the Implementation of the k-ε Turbulence Model in Incompressible Flow Solvers Based on a Finite Element Discretization," International Journal of Computing Science and Mathematics, vol.1, no. 2–4, pp. 193–206,

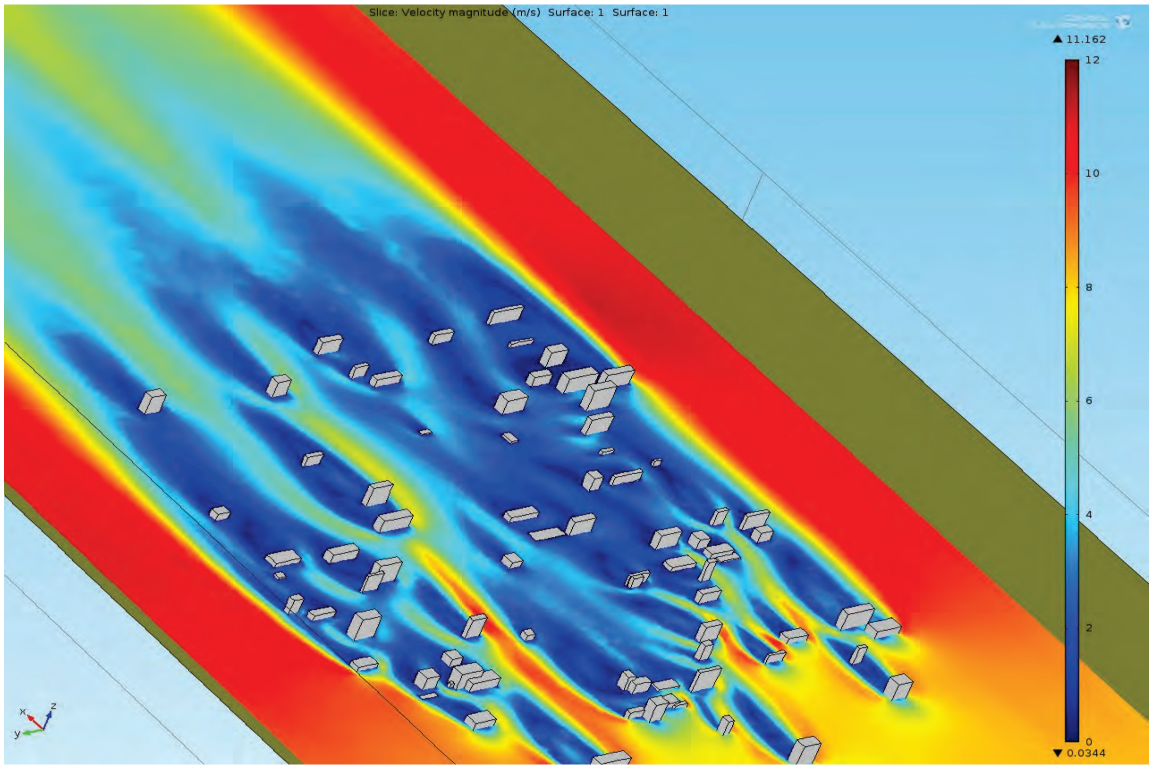
Wilcox, D.C., (1998). Turbulence Modeling for CFD, 2nd ed., DCW Industries, 1998.

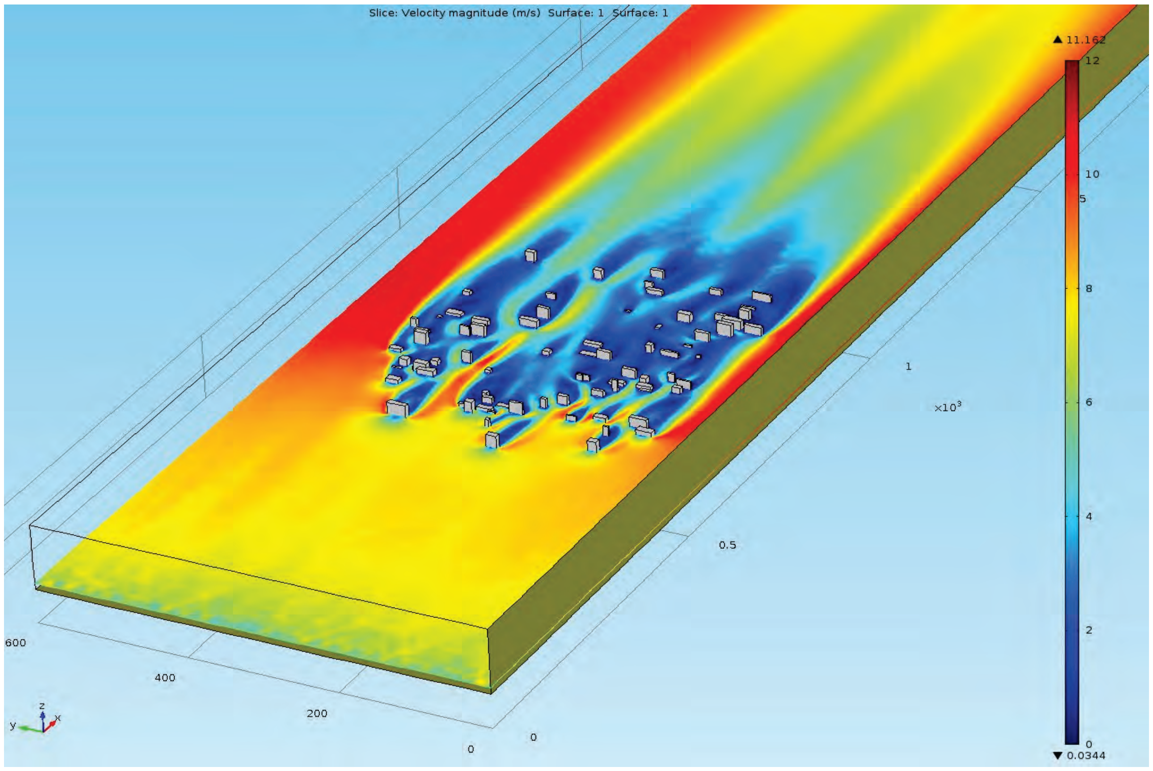
Appendix

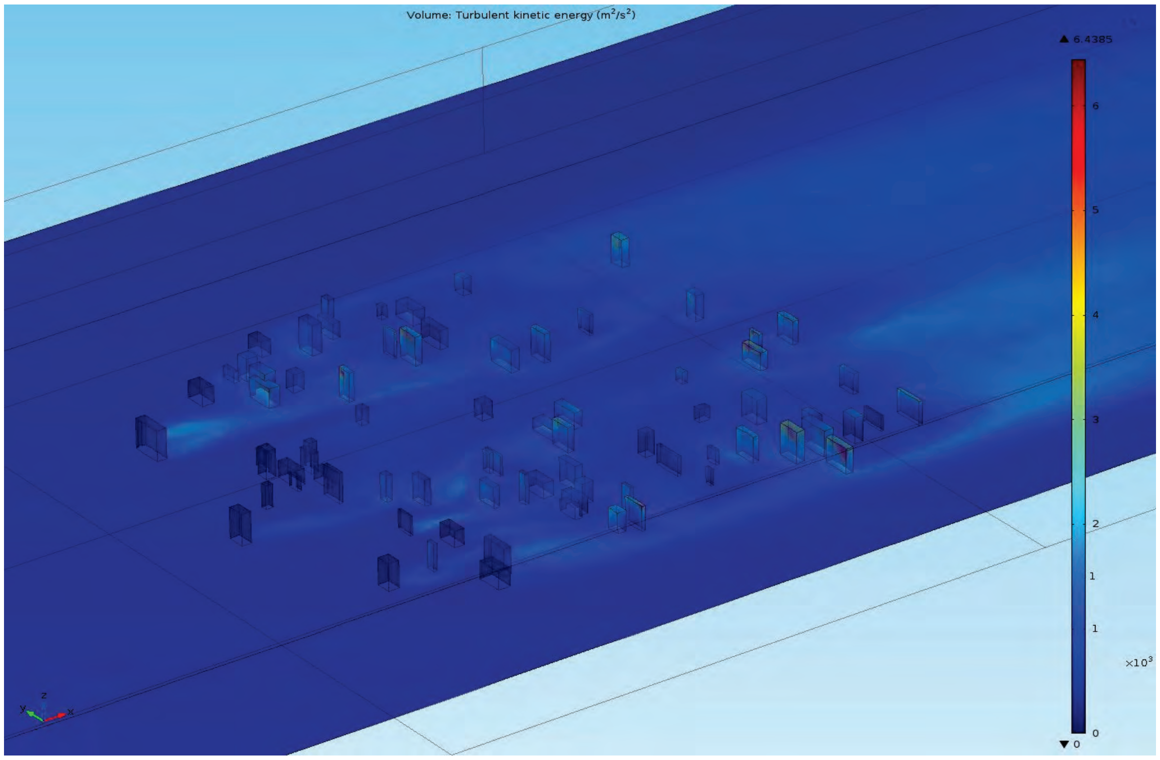
Matlab script (see Section 3)
Comsol results (see Section 4)

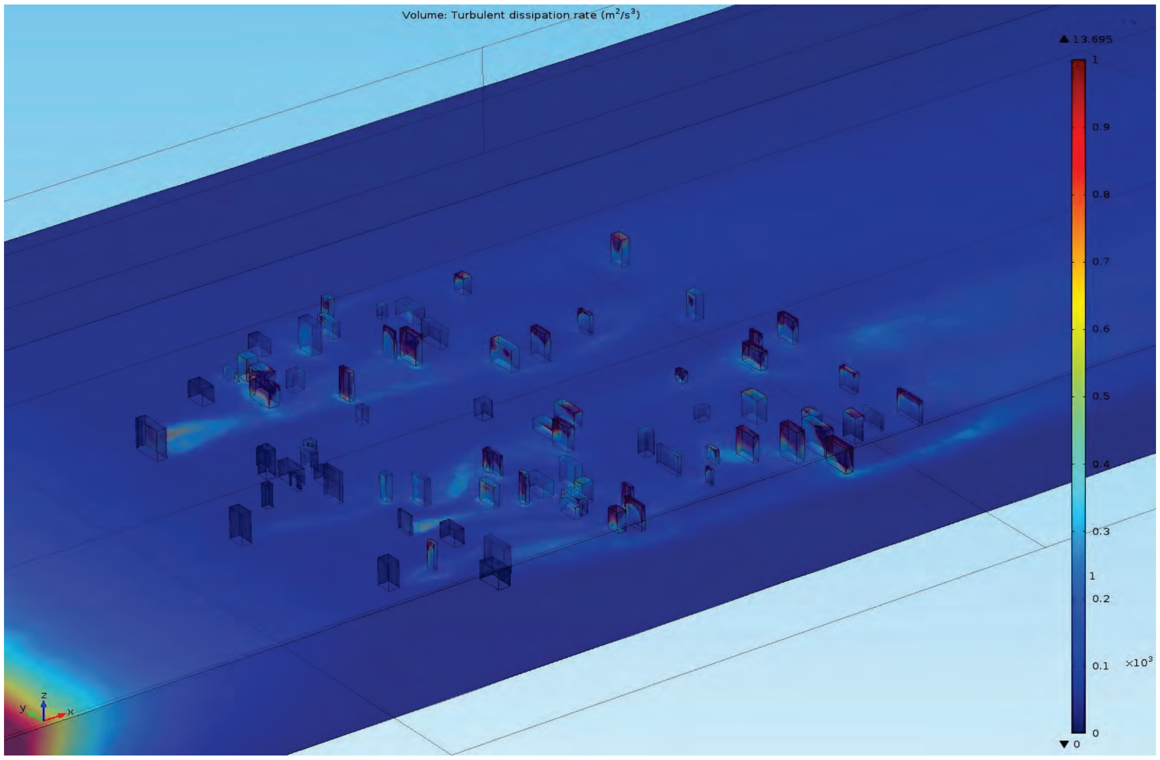
The MatLab Code for creating random urban area geometry in Comsol

```
import com.comsol.model.*
import com.comsol.model.util.*
model = ModelUtil.create('Model');
model.modelPath('D:\COMSOL42\mfiles');
model.name('MultiBuildingGeo1.mph');
model.modelNode.create('mod1');

model.geom.create('geom1', 3);
model.geom('geom1').feature.create('wp1', 'WorkPlane');
model.geom('geom1').feature('wp1').geom.feature.create('r1', 'Rectangle');
model.geom('geom1').feature.create('ext1', 'Extrude');
model.geom('geom1').feature('wp1').geom.feature('r1').set('size', {'2000' '600'});
model.geom('geom1').feature('ext1').setIndex('distance', '100', 0);
model.geom('geom1').feature('ext1').selection('input').set({'wp1.r1'});

nB=80;
xp=100+400*rand(nB,1);
yp=500+500*rand(nB,1);
sx=2+8*rand(nB,1);
sy=5+20*rand(nB,1);
hz=10+20*rand(nB,1);

uus=[];
for k=2:nB
ks=num2str(k,'%4.0f');
xps=num2str(xp(k),'%3.0f');
yps=num2str(yp(k),'%3.0f');
sixs=num2str(sx(k),'%3.0f');
siys=num2str(sy(k),'%3.0f');
hzs=num2str(hz(k),'%3.0f');
eval(['model.geom(' '''geom1''' ').feature(' '''wp1''' ').geom.feature.create(' '''' 'r' ks '''' ',' '''Rectangle''' ')']);
eval(['model.geom(' '''geom1''' ').feature.create(' '''' 'ext' ks '''' ',' '''Extrude''' ')']);
eval(['model.geom(' '''geom1''' ').feature(' '''wp1''' ').geom.feature(' '''' 'r' ks '''' ').set(' '''pos''' ', {' '''' yps '''' ' ' '''' xps '''' '})']);;
eval(['model.geom(' '''geom1''' ').feature(' '''wp1''' ').geom.feature(' '''' 'r' ks '''' ').set(' '''size''' ', {' '''' sixs '''' ' ' '''' siys '''' '})']);;
eval(['model.geom(' '''geom1''' ').feature(' '''' 'ext' ks '''' ').setIndex(' '''distance''' ',' '''' hzs '''' ',0)']);;
eval(['model.geom(' '''geom1''' ').feature(' '''' 'ext' ks '''' ').selection(' '''input''' ').set({' '''' 'wp1.r' ks '''' '}' ')']);
uus=[uus ' ' '''' 'ext' ks '''' ]
end

model.geom('geom1').feature.create('dif1', 'Difference');
model.geom('geom1').feature('dif1').selection('input').set({'ext1'});
eval(['model.geom(' '''geom1''' ').feature(' '''dif1''' ').selection(' '''input2''' ').set({' uus '}' ')']);

model.geom('geom1').run;
model.view('view1').hideEntities.create('hide1');
model.view('view1').hideEntities('hide1').geom('geom1', 2);
model.view('view1').hideEntities('hide1').set([4]);
model.material.create('mat1');
model.material('mat1').propertyGroup('def').func.create('eta', 'Piecewise');
model.material('mat1').propertyGroup('def').func.create('Cp', 'Piecewise');
model.material('mat1').propertyGroup('def').func.create('rho', 'Analytic');
model.material('mat1').propertyGroup('def').func.create('k', 'Piecewise');
model.material('mat1').propertyGroup('def').func.create('cs', 'Analytic');
model.material('mat1').selection.set([1]);
model.mesh.create('mesh1', 'geom1');
model.mesh('mesh1').feature.create('ftet1', 'FreeTet');
model.view('view1').set('renderwireframe', true);
model.view('view1').set('transparency', 'on');
model.mesh('mesh1').feature('size').set('table', 'cfd');
model.mesh('mesh1').feature('size').set('hauto', 7);
model.mesh('mesh1').run;
model.mesh('mesh1').run;
mphsave(model,'MBtest6')
```


Slice: Velocity magnitude (m/s) Surface: 1 Surface: 1
▲ 11.162
12
10
8
6
4
2
0
▼ 0.0344


Slice: Velocity magnitude (m/s) Surface: 1 Surface: 1
▲ 11.162
12
10
8
6
4
2
0
▼ 0.0344
1
×10³
0.5
0
600
400
200
0


Volume: Turbulent kinetic energy (m²/s²)
▲ 6.4385
6
5
4
3
2
1
1
0
×10³
▼ 0


Volume: Turbulent dissipation rate (m²/s³)
▲ 13.695
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
×10³
▼ 0