# QES-Winds v1.0: Theory and User's Guide

Behnam Bozorgmehr[1], Pete Willemsen[2], Fabien Margairaz[1], Jeremy Gibbs[1,3,4], Zachary Patterson[2], Rob Stoll[1], and Eric Pardyjak[1]

[1]Department of Mechanical Engineering, University of Utah
[2]Department of Computer Science, University of Minnesota Duluth
[3]Cooperative Institute for Mesoscale Meteorological Studies, University of Oklahoma
[4]NOAA/OAR/National Severe Storms Laboratory, Norman, Oklahoma

January 2021

# Contents

## 1 Introduction

A new dispersion modeling system based on the well-used FORTRAN-based QUIC (Quick Urban and Industrial Complex) dispersion modeling system originally developed by the University of Utah and Los Alamos National Laboratory [1], has been under development as collaboration between the University of Utah and the University of Minnesota, Duluth. Quick Environmental Simulation (QES) is a microclimate simulation platform for computing 3D environmental scalars in urban areas and over complex terrain. Figure 1 shows a schematic of QES system and how different elements of the system interact with each other.

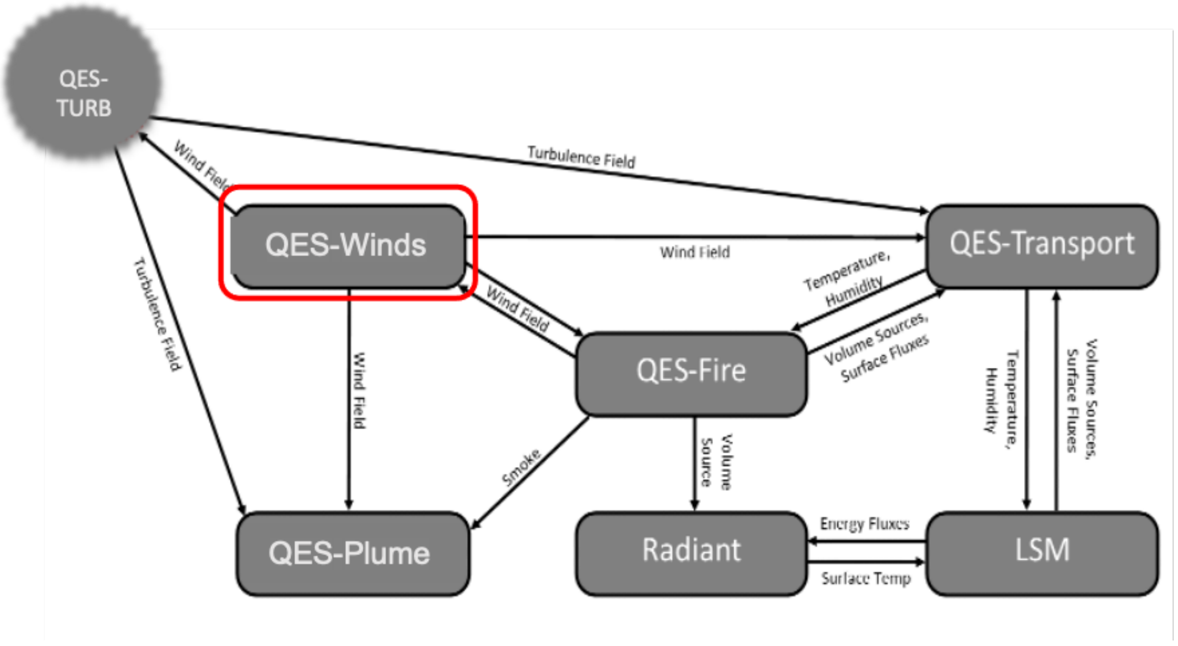
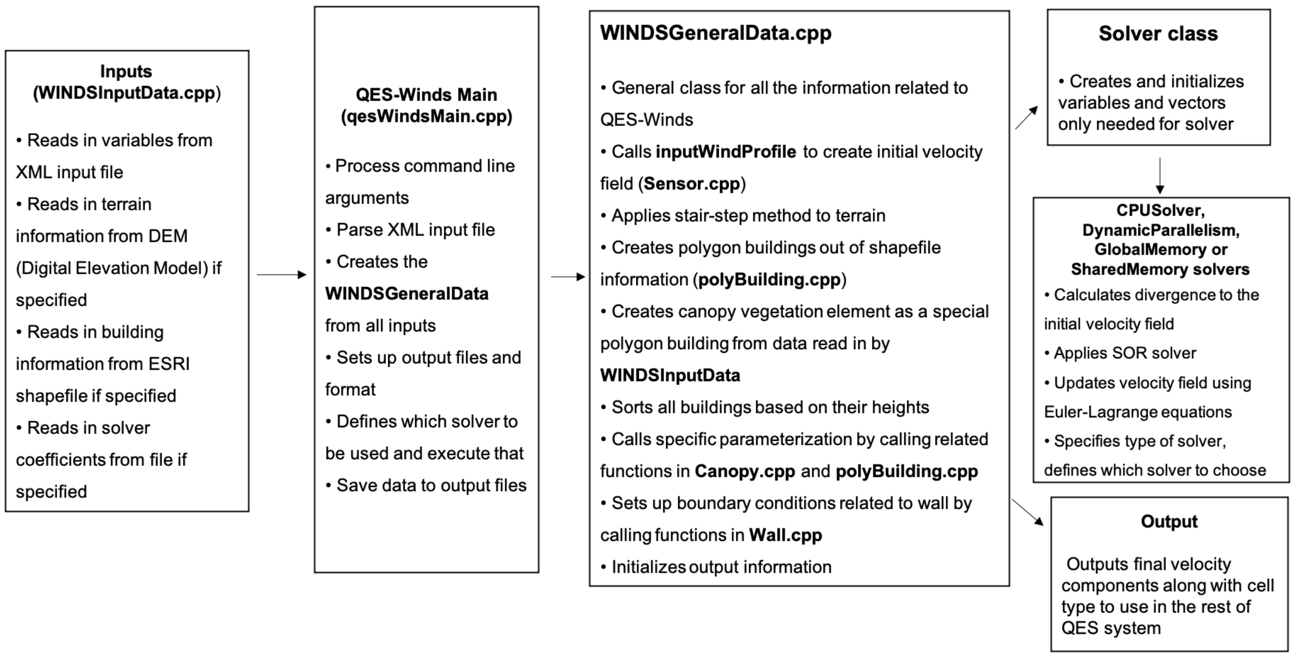


Figure 1: Schematic of the Quick Environmental System (QES) and the relationship between different elements of the system including data flow from one element to the other.

The fast-response three-dimensional diagnostic wind model written in C++, QES-Winds, is a rapid mass conserving wind-field solver. QES-Winds utilizes the concept of dynamic parallelism in CUDA to substantially accelerate wind simulations. Figure 2 shows a high-level flowchart for QES-Winds code.



Inputs (WINDSInputData.cpp)
• Reads in variables from XML input file
• Reads in terrain information from DEM (Digital Elevation Model) if specified
• Reads in building information from ESRI shapefile if specified
• Reads in solver coefficients from file if specified
QES-Winds Main (qesWindsMain.cpp)
• Process command line arguments
• Parse XML input file
• Creates the WINDSGeneralData from all inputs
• Sets up output files and format
• Defines which solver to be used and execute that
• Save data to output files
WINDSGeneralData.cpp
• General class for all the information related to QES-Winds
• Calls inputWindProfile to create initial velocity field (Sensor.cpp)
• Applies stair-step method to terrain
• Creates polygon buildings out of shapefile information (polyBuilding.cpp)
• Creates canopy vegetation element as a special polygon building from data read in by WINDSInputData
• Sorts all buildings based on their heights
• Calls specific parameterization by calling related functions in Canopy.cpp and polyBuilding.cpp
• Sets up boundary conditions related to wall by calling functions in Wall.cpp
• Initializes output information
Solver class
• Creates and initializes variables and vectors only needed for solver
CPUSolver, DynamicParallelism, GlobalMemory or SharedMemory solvers
• Calculates divergence to the initial velocity field
• Applies SOR solver
• Updates velocity field using Euler-Lagrange equations
• Specifies type of solver, defines which solver to choose
Output
Outputs final velocity components along with cell type to use in the rest of QES system

Figure 2: Flowchart for the QES-Winds wind solver

## 2 QES-Winds Domain

The first step in every computational code or package is to define the computational domain. The user can define the domain by specifying the number of cells in $x$, $y$ and $z$ directions as well as the cell size in each direction in the input file (XML file).

### 2.1 XML Example

The domain information (number of cells and cell size) are defined under the <simulationParameters> part of the XML file. Following is an example of a domain with 2 km by 2 km by 200 m and resolution of 2 m by 2 m by 2 m:

```
<simulationParameters>
   <domain> 1000 1000 100 </domain>     <!-- Number of cells in x,y and z directions-->
   <cellSize> 2.0 2.0 2.0 </cellSize>     <!-- Mesh resolution (meters)-->
</simulationParameters>
```

## 3 Staggered Grid

QES-Winds discretizes the computational domain using a staggered grid as shown in Figure 3. The velocity components ($u$, $v$ and $w$) are face-centered values and Lagrange multipliers ($\lambda$), divergence of the initial wind field ($R$) and solver coefficients ($e$, $f$, $g$, $h$, $m$ and $n$) are cell-centered variables. Because of nature of the finite difference method (depending on neighboring cells values), the first and last cells in $x$ and $y$ directions and the last cell in $z$ direction, are not updated in the solving process (their velocity remains as the same as the initial velocity field). For the same reason, there is a layer of ghost cells under the bottom surface to make velocity calculation in the first layer above the bottom surface possible. The values of the Lagrange multipliers for the ghost cells are set to the ones for the layer above the bottom surface to create a zero gradient for the Lagrange multipliers (boundary condition) as well as providing the neighboring cell for the finite difference method.

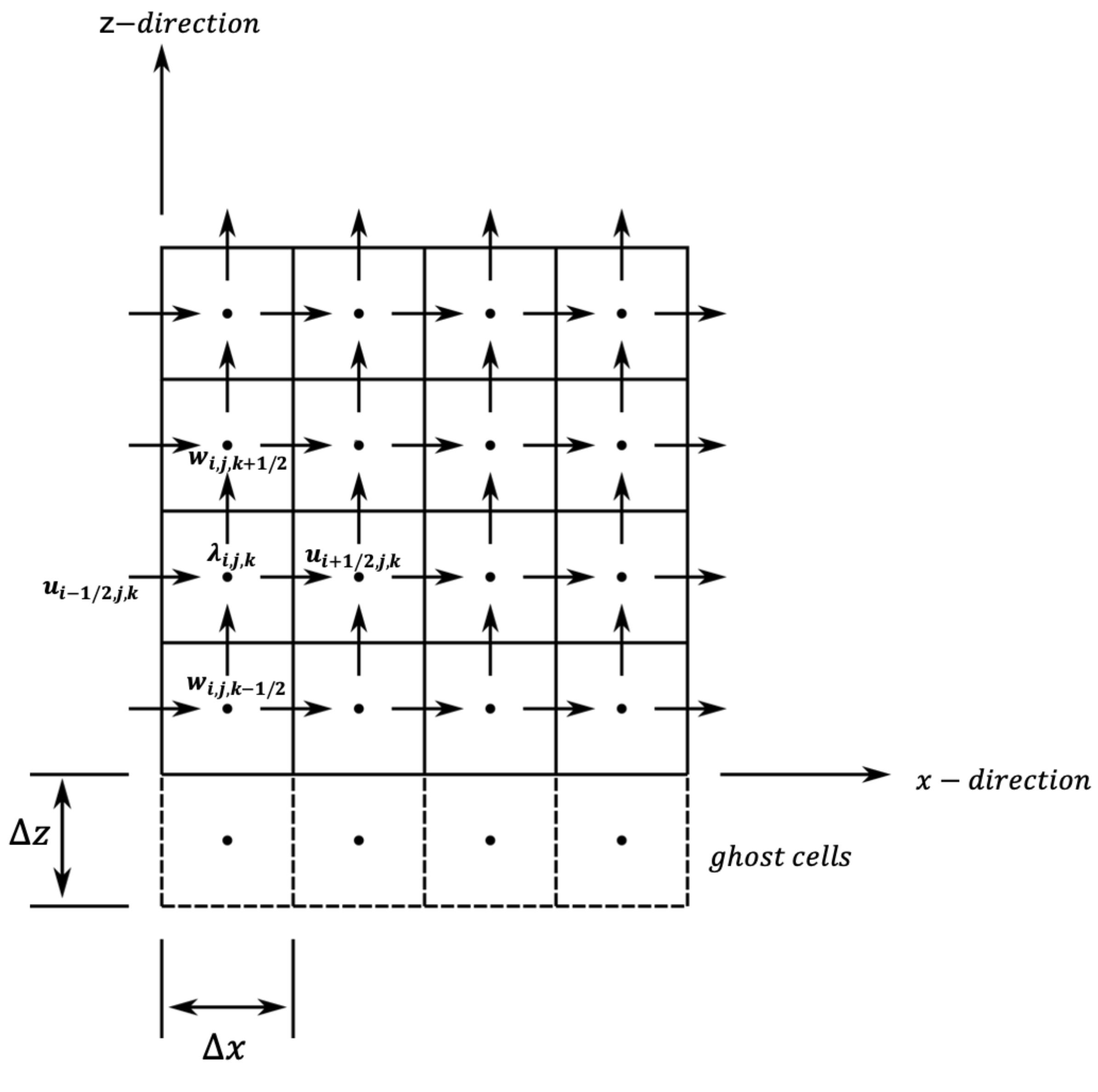


Figure 3: Staggered grid representation of the domain and location of each variable.

## 3.1 Halo Region

If a solid element (building or terrain) overlaps with the QES domain boundaries, QES-Winds cannot model the wind field around the element correctly. In order to prevent this phenomenon, the user can add buffer zones to the sides of the domain when a terrain file or an ESRI shpefile is read into the code. Figure 4 represents how the halo region is added to the domain around a Digital Elevation Model (DEM) or a shapefile.

In order to define length of the halo zone in $x$ and $y$ direction, the user can use <halo_x> and <halo_y> under <simulationParameters>. When the halo zone is defined, the length of the domain

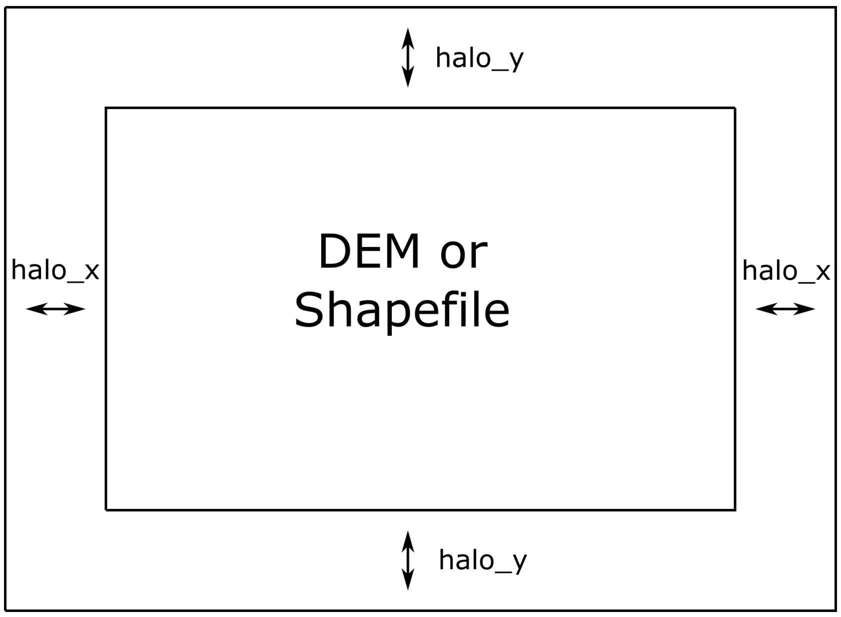


Figure 4: Representation of halo region around the domain.

$(nx * dx)$ and $(ny * dy)$, should be greater than or equal to length of the DEM or shapefile in each direction plus twice the length of the halo in $x$ and $y$ directions, respectively.

```
<simulationParameters>
    <halo_x> 20.0 </halo_x>      <!-- Halo region added to x-direction of domain (at the
        beginning and the end of domain) (meters)-->
    <halo_y> 30.0 </halo_y>      <!-- Halo region added to y-direction of domain (at the
        beginning and the end of domain) (meters)-->
</simulationParameters>
```

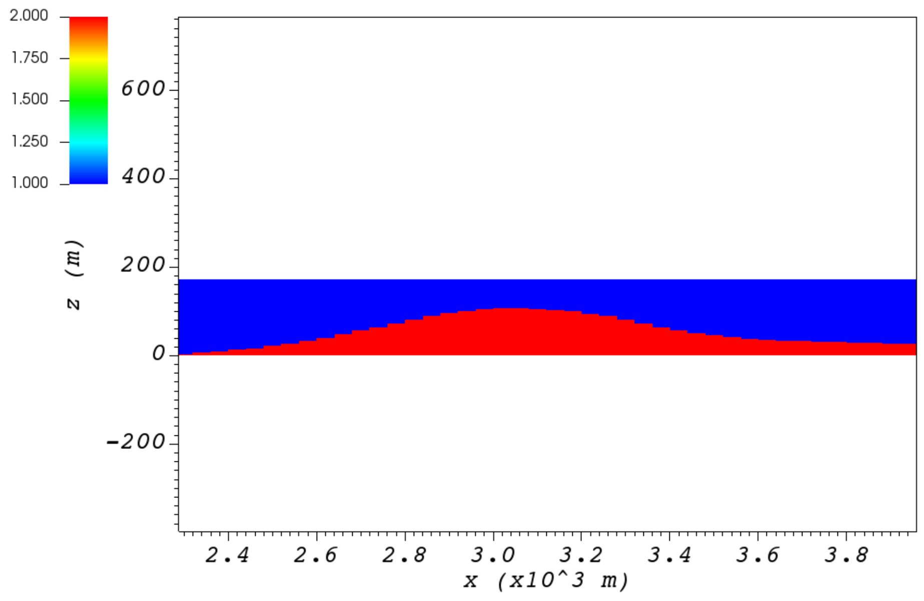
## 4 Digital Elevation Model (DEM) and ESRI Shapefile

The current version of QES-Winds has been written to allow commonly available terrain and building geometry datasets to be used for simulations. In this section, various input file formats for QES-Winds will be covered.

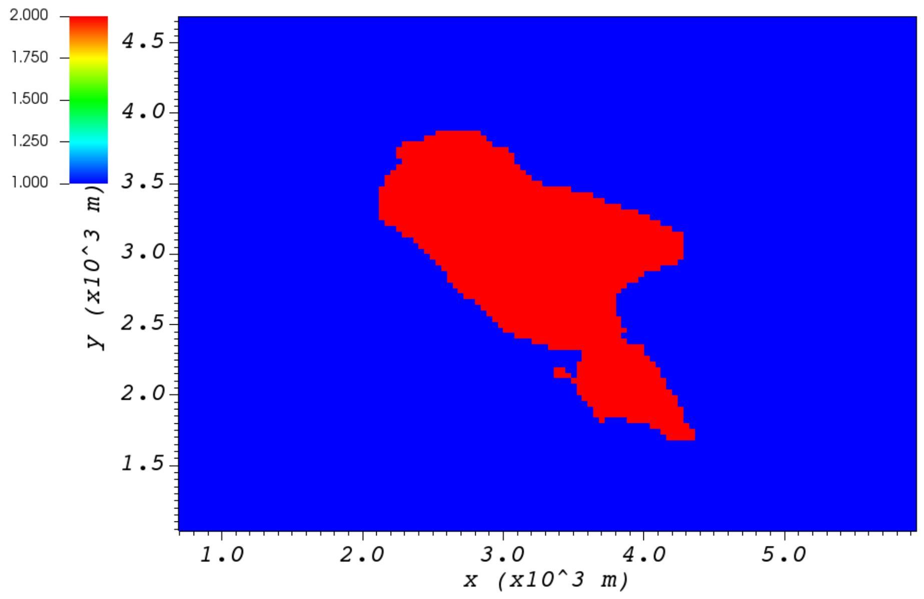
### 4.1 Terrain Features

Using the Geospatial Data Abstraction Library (GDAL; https://www.gdal.org), we are able to load geo-referenced datasets of terrain so that the simulations can include the effects of hills, valleys, and mountains. In the current version of the code, we can load Digital Elevation Model (DEM) files for different physical locations.

Using the Digital Elevation Model (DEM) file loaders in our code base, we have loaded and tested multiple different terrain data sets. As a first test, we loaded a DEM of Askervein Hill. This is an isolated hill in Scotland where field experiments have been conducted and data for testing and evaluation exists ( [2,3]). The simulation with Askervein Hill was run without any complex terrain flow parameterizations. The Askervein Hill dataset is 6023.43 m by 6023.43 m. The hill height is approximately 124 m tall. Figure 5 indicates the cell type contour for the Askervin hill test case in a vertical plane at $y = 3000$ m (part (a)), and a horizontal plane at $z = 20$ m (part (b)). These plots show the ability of QES-Winds to read in and process DEM files. The cell type value 1 (blue) represents the air cells while value 2 (red) indicates the terrain cells.



(a)



(b)

Figure 5: Cell type contour for the Askervin hill test case in a (a) vertical plane at $y$ = 3000 m, (b) horizontal plane at $z$ = 20 m. The cell type value 1 (blue) represents the air cells while value 2 (red) indicates the terrain cells.

The user can define the address to the DEM using <DEM> variable under the <simulationParameters> part in the XML file:

```
<simulationParameters>
    <DEM>../scratch/DEM/askervein.tif</DEM>    <!-- Address to DEM location-->
</simulationParameters>
```

### 4.1.1 Process Part of DEM

In some cases, user wants to load a giant DEM but only process part of the file. This is possible in QES-Winds by defining the origin of QES domain inside the DEM borders and the size of the QES domain. Figure 6 shows a schematic of how the QES domain can be defined inside a DEM file and only process that part.

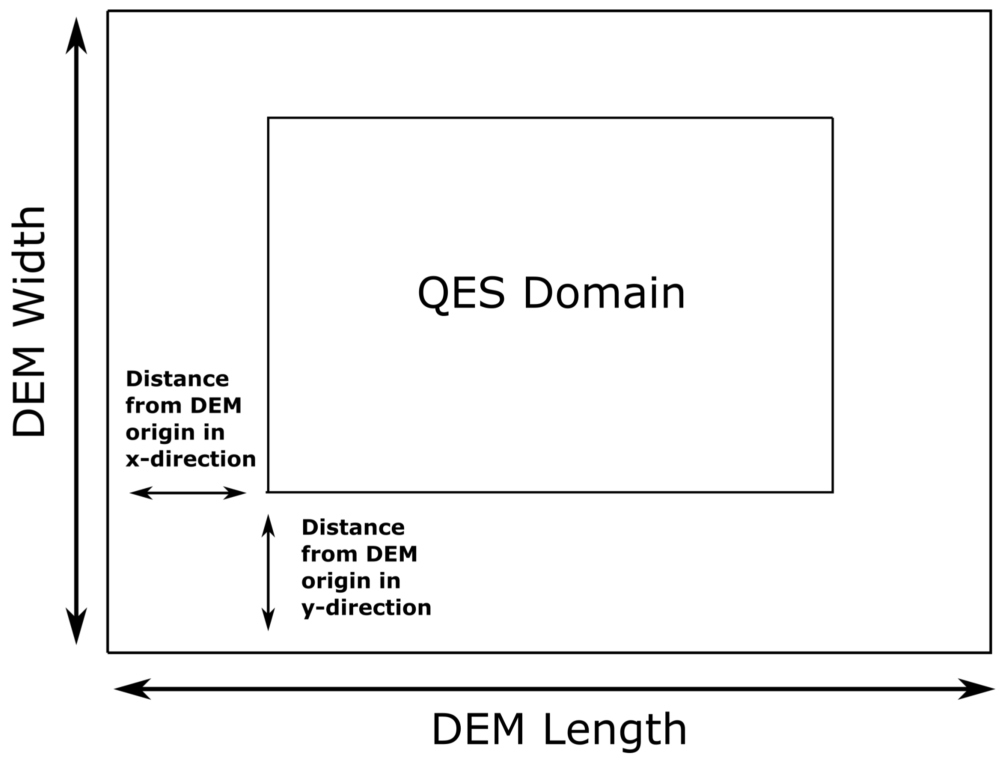

DEM Width
QES Domain
Distance from DEM origin in x-direction
Distance from DEM origin in y-direction
DEM Length

Figure 6: Schematic of how the QES domain can be defined inside a DEM file and only process that part.

There are two options to determine the location of the origin of QES domain inside the DEM borders: 1) Specifying the distance of the QES origin with respect to bottom left corner of the DEM file. This can be done by setting the value of <originFlag> to 0 and defining distances (in meters) in $x$ and $y$ directions using <DEMDistancex> and <DEMDistancey>, respectively.

```
<simulationParameters>
    <originFlag> 0 </originFlag>     <!-- Origin flag (0- DEM coordinates (default), 1- UTM
        coordinates) -->
    <DEMDistancex> 1000.0 </DEMDistancex>     <!-- x component (m) of origin in DEM coordinates
        (if originFlag = 0) -->
    <DEMDistancey> 1000.0 </DEMDistancey>     <!-- y component (m) of origin in DEM coordinates
        (if originFlag = 0) -->
</simulationParameters>
```

2) Defining the location of the QES domain origin in the Universal Transverse Mercator (UTM) coordinates by setting the value of <originFlag> to 1 and determining <UTMx> and <UTMy> of the origin in $x$ and $y$ directions, respectively.

```
<simulationParameters>
   <originFlag> 1 </originFlag>     <!-- Origin flag (0- DEM coordinates (default), 1- UTM
       coordinates) -->
   <UTMx> 595469.6122881 </UTMx>      <!-- x component (m) of origin in UTM DEM coordinates (if
        originFlag = 1)-->
   <UTMy> 6336281.9538635 </UTMy>      <!-- y component (m) of origin in UTM DEM coordinates (
       if originFlag = 1)-->
</simulationParameters>
```

## 4.2 Automated City Building

A new shapefile reader function has been added to QES-Winds, which provides the capacity to load the ESRI shapefiles using GDAL (Geospatial Data Abstraction Library) libraries. After the building footprints and heights are loaded from ESRI shapefiles, QES-Winds creates polygon buildings and applies appropriate parameterization to them. Figure 7 shows an example ESRI shapefile can be read into QES-Winds, Central Business District (CBD) of Oklahoma City shapefile, subject to JU2003 experimental campaign [4], plotted using the freely available software QGIS (https://qgis.orgg). The cell type contour for the Oklahoma City test case in a horizontal plane at $z = 3$ m is shown in Figure 8. This plot indicates the ability of QES-Winds to read in and process ESRI shapefiles. The cell type value 0 (blue) represents the building cells while value 1 (red) indicates the air cells.

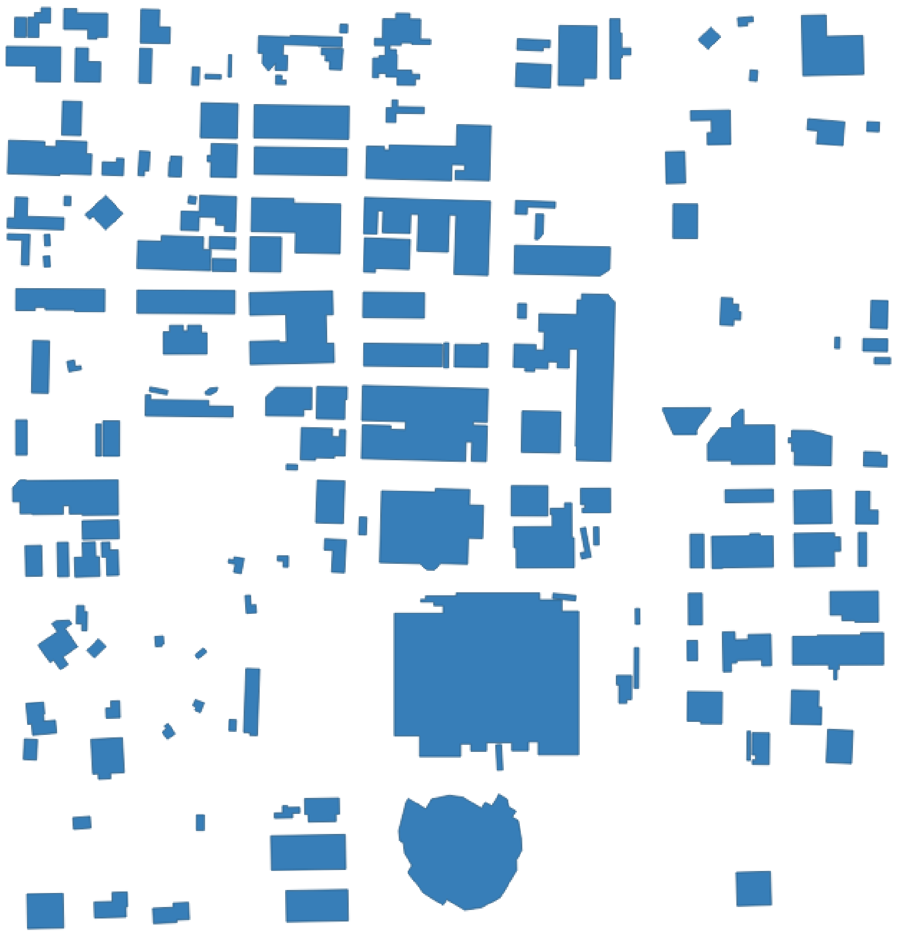


Figure 7: Central Business District (CBD) of Oklahoma City shapefile, subject to JU2003 experimental campaign [4], plotted using the freely available software QGIS.

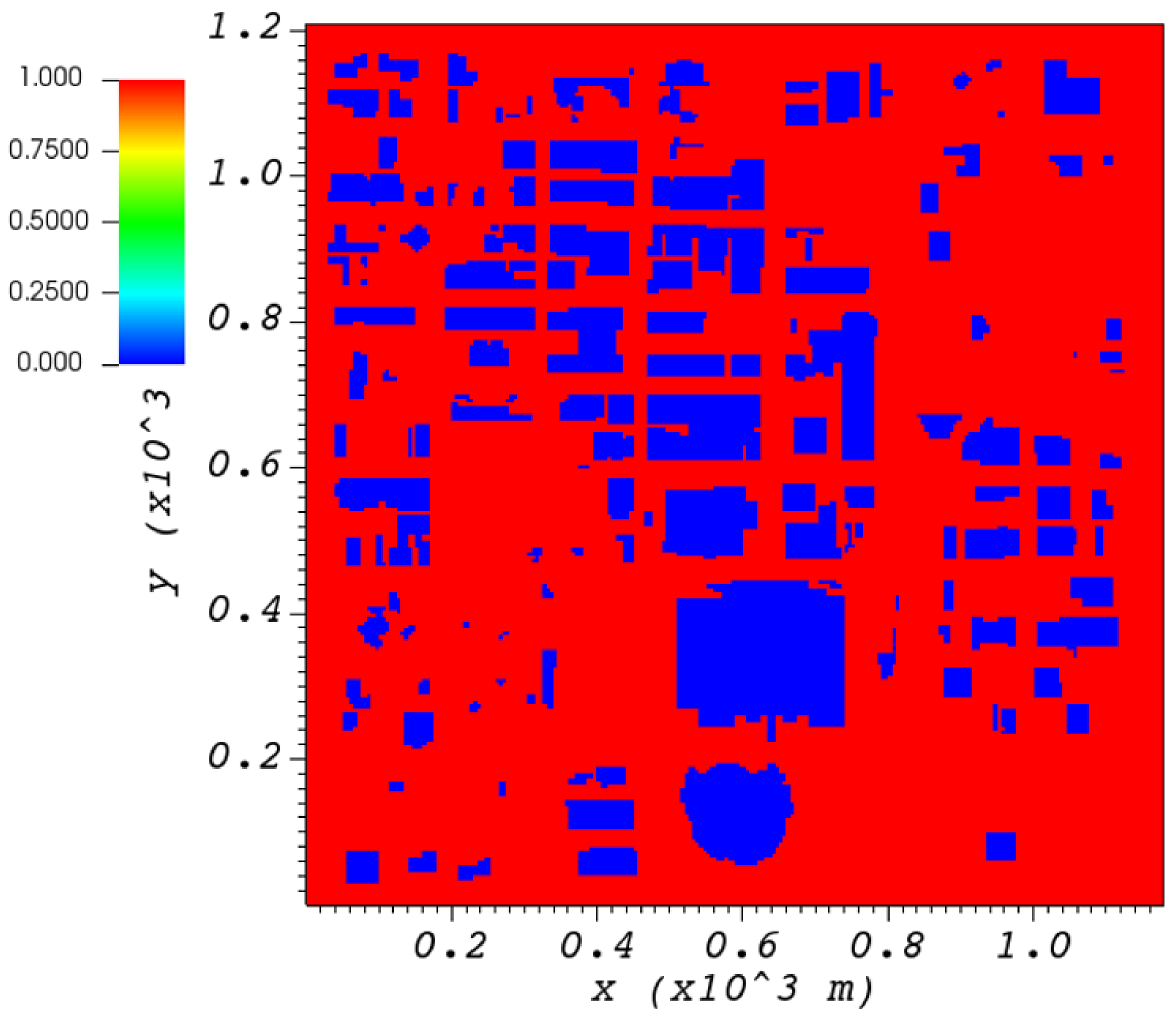


Figure 8: Cell type contour for the Oklahoma City test case in a horizontal plane at $z = 3$ m. The cell type value 0 (blue) represents the building cells while value 1 (red) indicates the air cells.

The user can define the address to the shapefile using <SHP> variable as well as the name of the shapefile using the <SHPBuildingLayer> and the correlation factor between the height field of the shapefile and the actual height of the buildings using the <heightFactor> under <simulationParameters> part in the XML file:

```
<simulationParameters>
    <SHP>../data/GISFiles/OKCSmallDomain/OKCSmallDomainJU2003.shp</SHP> <!-- Address to
        shapefile location-->
    <SHPBuildingLayer>OKCSmallDomainJU2003</SHPBuildingLayer>
    <heightFactor> 1.0 </heightFactor>    <!-- Height factor multiplied by the building height
        read in from the shapefile (default = 1.0)-->
</simulationParameters>
```

### 4.3 Import Building From XML

Instead of reading in a ESRI shapefile, the user can import building information manually through the XML file. This can be done by using the <buildings> section of the XML file. The only option available for now is the rectangular building. Information required for defining a rectangular building are height, base height, length, width, location of the closest corner to the origin of domain and building rotational angle. Following is an example of a rectangular building with 40 m as height, 0m as base height, 20 m as length and width, closest corner to the origin located at 90 m in $x$ and $y$ directions, and 0 ° as rotation angle with respect to the North-South line. Also, 0.1 m is defined as the surface roughness for all the building walls.

```
<buildings>
  <wallRoughness> 0.1 </wallRoughness>
  <rectangularBuilding>
    <height> 40.0 </height>
    <baseHeight> 0 </baseHeight>
    <xStart> 90.0 </xStart>
    <yStart> 90.0 </yStart>
    <length> 20.0 </length>
    <width> 20.0 </width>
    <buildingRotation> 0.0 </buildingRotation>
  </rectangularBuilding>
</buildings>
```

## 5 Initial Wind Field

QES-Winds can read a single or multiple sensors for a specific test case. In this context, sensor means the velocity magnitude and direction at a single point or a single velocity profile to initialize the wind field. If there is only the wind velocity and direction at a single point, the user should specify what type of velocity profile they want to build from the measurement. There are three options available for the type of profile: 1) a logarithmic profile (Eq. 1) [5], 2) a power law profile (Eq. 2) [5] and 3) an urban canopy profile (Eq. 3, 4) [5, 6].

$$u_{log}(z) = u_{ref} \cdot \frac{ln(z/z_0)}{ln(z_{ref}/z_0)} \tag{1}$$

$$u_{pow}(z) = u_{ref}.(z/z_{ref})^{exp} \tag{2}$$

$$u_{uc}(z \leq H) = u(H).exp(\alpha(\frac{z}{H} - 1)) \tag{3}$$

$$u_{uc}(z > H) = \frac{u_*}{\kappa}.ln(\frac{z-d}{z_0}) \tag{4}$$

where $u_{ref}$ is the measured velocity at measured height $z_{ref}$, $z_0$ is the surface roughness. In the power law profile shown in Eq. 2, $exp$ is the velocity profile exponent. The lower portion of the urban canopy profile calculated in Eq. 3 where $\alpha$ is a factor that depends on canopy element density (attenuation coefficient) and $u(H)$ is the computed velocity at height $H$. The upper portion of the urban canopy is a different form of a logarithmic profile where $u_*$ is the friction velocity, $\kappa$ is the von Karman constant at 0.4 and $d$ is the zero plane displacement.

If there is only one sensor available in the computational domain, the code will extend the profile for that sensor uniformly to the whole domain. On the occasion of multiple sensors, QES-Winds utilizes a two-dimensional Barnes interpolation scheme [7, 8] to interpolate velocity components at each cell height of the domain based on the weighted distance from each sensor.

## 5.1 XML Setup

There are two options available for defining sensor information: 1) the user can put all the sensor information in a separate XML file and define the address to the location of the sensor file using the <sensorName> variable.

```
<metParams>
  <z0_domain_flag> 0 </z0_domain_flag>    <!-- Distribution of surface roughness for domain (0-
      uniform (default), 1-custom -->
  <sensorName>../data/InputFiles/sensor.xml</sensorName> <!-- Name of the sensor file with
      information for the sensor included -->
</metParams>
```

2) The user can define all information required for creating a sensor by using the <sensor> variable inside the <metParams> section of the XML file.

The first part of the sensor information is the location of the sensor in domain. There are three options for it: 1) define the location in local coordinates of the QES domain.

```
<metParams>
  <sensor>
      <site_coord_flag> 1 </site_coord_flag>    <!-- Sensor site coordinate system (1=QES (
          default), 2=UTM, 3=Lat/Lon) -->
      <site_xcoord> 1.0 </site_xcoord>     <!-- x component of site location in QES domain (m) (
          if site_coord_flag = 1) -->
      <site_ycoord> 1.0 </site_ycoord>     <!-- y component of site location in QES domain (m) (
          if site_coord_flag = 1)-->
  </sensor>
</metParams>
```

2) The user can define the location in the Universal Transverse Mercator (UTM) coordinates. In this case, user also needs to define the origin of computational domain in the UTM coordinates.

```
<simulationParameters>
  <UTMx> 634173 </UTMx>       <!-- x component (m) of origin in UTM -->
  <UTMy> 3925360 </UTMy>       <!-- y component (m) of origin in UTM -->
  <UTMZone> 14 </UTMZone>       <!-- UTM zone that domain located -->
</simulationParameters>
```

```
<metParams>
  <sensor>
      <site_coord_flag> 2 </site_coord_flag>    <!-- Sensor site coordinate system (1=QES (
          default), 2=UTM, 3=Lat/Lon) -->
      <site_UTM_x> 634175 </site_UTM_x>     <!-- x components of site coordinate in UTM (if
          site_coord_flag = 2) -->
      <site_UTM_y> 3925362 </site_UTM_y>     <!-- y components of site coordinate in UTM (if
          site_coord_flag = 2)-->
      <site_UTM_zone> 14 </site_UTM_zone>     <!-- UTM zone of the sensor site (if
          site_coord_flag = 2)-->
  </sensor>
</metParams>
```

3) The user can define the location in Latitude and Longitude coordinates. In this case, user also needs to define the origin of computational domain in the UTM coordinates.

```
<simulationParameters>
  <UTMx> 634173 </UTMx>       <!-- x component (m) of origin in UTM -->
  <UTMy> 3925360 </UTMy>       <!-- y component (m) of origin in UTM -->
  <UTMZone> 14 </UTMZone>       <!-- UTM zone that domain located -->
</simulationParameters>
```

```
<metParams>
  <sensor>
      <site_coord_flag> 3 </site_coord_flag>    <!-- Sensor site coordinate system (1=QES (
          default), 2=UTM, 3=Lat/Lon) -->
      <site_lat> 35.46270 </site_lat>     <!-- x components of site coordinate in Latitude (if
          site_coord_flag = 3) -->
      <site_lat> -97.52130 </site_lat>     <!-- y components of site coordinate in Longitude (if
           site_coord_flag = 3)-->
  </sensor>
</metParams>
```

The second part of sensor definition is choosing type of profile for different time steps, if applicable. The <timeSeries> variable is designed to define type of sensor profile in the sensor section for several time steps. There are four options for the input profile in QES-Winds: 1) Logarithmic velocity profile, based on Eq. 1:

```
<metParams>
  <sensor>
      <timeSeries>      <!-- Start of timestep informastion for a sensor -->
          <boundaryLayerFlag> 1 </boundaryLayerFlag>   <!-- Site boundary layer flag (1-log (
              default), 2-exp, 3-urban canopy, 4-data entry) -->
          <siteZ0> 0.1 </siteZ0>      <!-- Site z0 -->
          <reciprocal> 0.0 </reciprocal>     <!-- Reciprocal Monin-Obukhov Length (1/m) -->
          <height> 20.0 </height>     <!-- Height of the sensor -->
          <speed> 5.0 </speed>      <!-- Measured speed at the sensor height -->
          <direction> 270.0 </direction>     <!-- Wind direction of sensor -->
       </timeSeries>
  </sensor>
</metParams>
```

Figure 9 shows velocity magnitude contour with overlaying velocity vectors of initial velocity field created by the aforementioned example of the logarithmic profile.

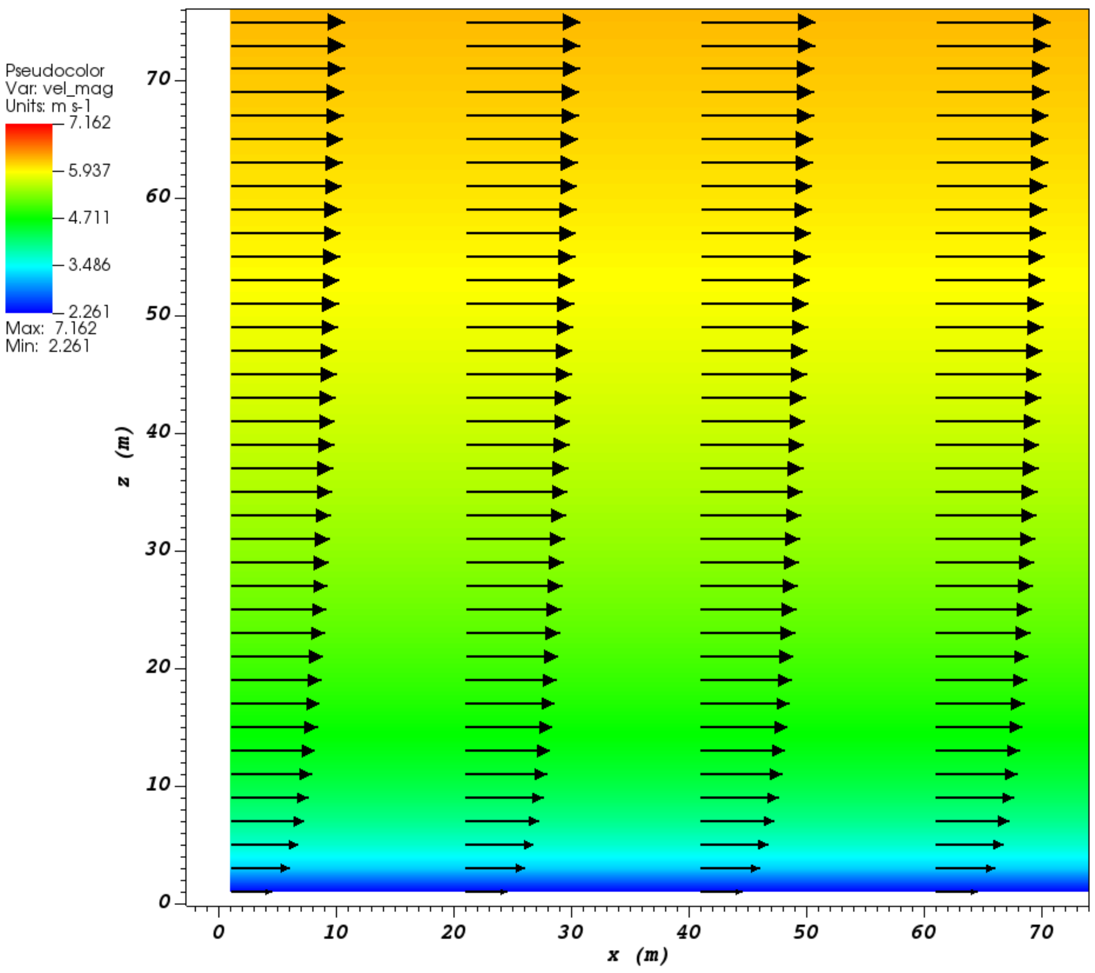


Figure 9: Velocity magnitude contour with overlaying velocity vectors in a vertical plane at $y = 101$ m for initial velocity field created by the logarithmic profile.

2) Exponential (power law) velocity profile, based on Eq. 2:

```
<metParams>
  <sensor>
      <timeSeries>      <!-- Start of timestep informastion for a sensor -->
          <boundaryLayerFlag> 2 </boundaryLayerFlag>   <!-- Site boundary layer flag (1-log (
              default), 2-exp, 3-urban canopy, 4-data entry) -->
          <siteZ0> 0.1 </siteZ0>      <!-- Site z0 -->
          <reciprocal> 0.0 </reciprocal>     <!-- Reciprocal Monin-Obukhov Length (1/m) -->
          <height> 20.0 </height>     <!-- Height of the sensor -->
          <speed> 5.0 </speed>      <!-- Measured speed at the sensor height -->
          <direction> 270.0 </direction>     <!-- Wind direction of sensor -->
       </timeSeries>
  </sensor>
```

```
</metParams>
```

Figure 10 shows velocity magnitude contour with overlaying velocity vectors of the initial velocity field created by the aforementioned example of the exponential (power law) profile.

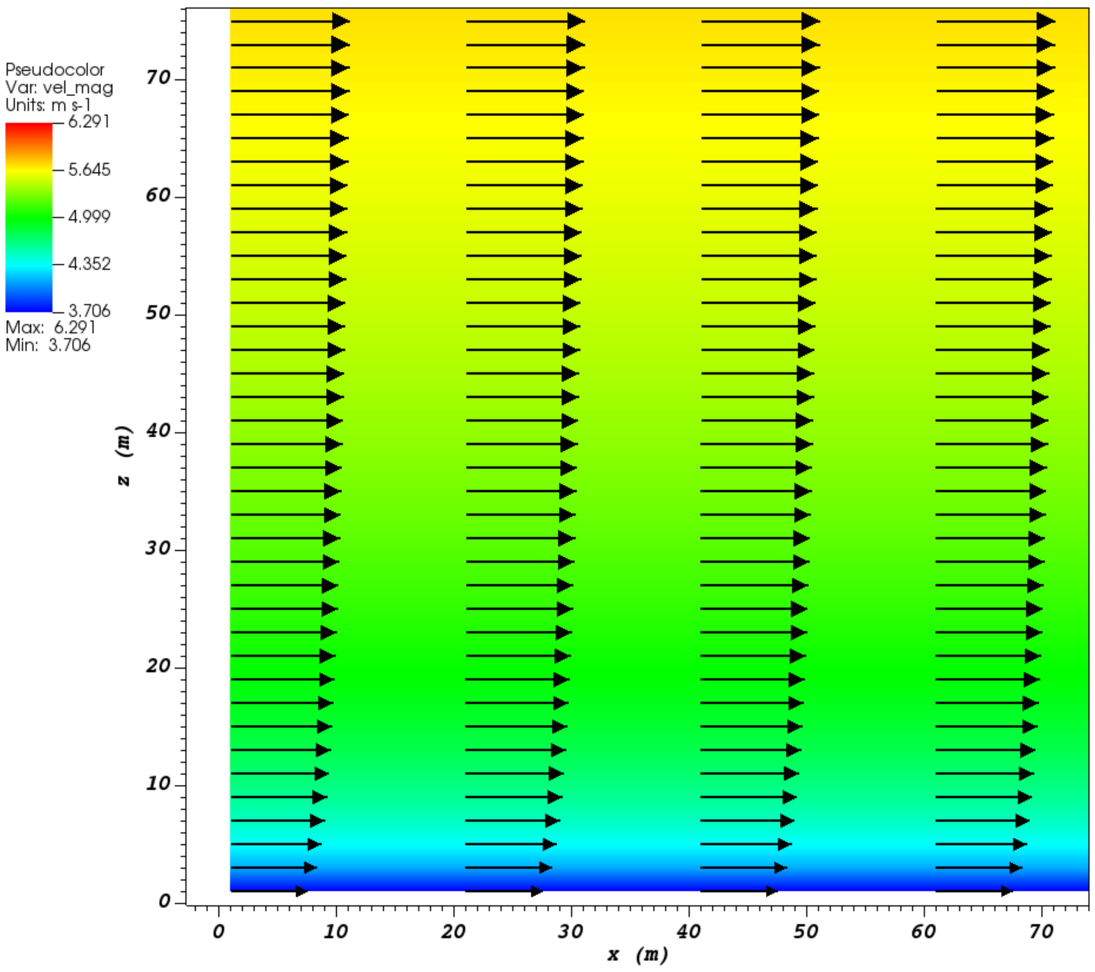


Figure 10: Velocity magnitude contour with overlaying velocity vectors in a vertical plane at $y$ = 101 m for initial velocity field created by the exponential (power law) profile.

3) Urban canopy velocity profile, based on Eq. 3 and 4:

```
<metParams>
  <sensor>
      <timeSeries>      <!-- Start of timestep informastion for a sensor -->
          <boundaryLayerFlag> 3 </boundaryLayerFlag>   <!-- Site boundary layer flag (1-log (
               default), 2-exp, 3-urban canopy, 4-data entry) -->
          <siteZ0> 0.1 </siteZ0>      <!-- Site z0 -->
          <reciprocal> 0.0 </reciprocal>     <!-- Reciprocal Monin-Obukhov Length (1/m) -->
```

```
            <height> 20.0 </height>      <!-- Height of the sensor -->
            <speed> 5.0 </speed>      <!-- Measured speed at the sensor height -->
            <direction> 270.0 </direction>     <!-- Wind direction of sensor -->
            <canopyHeight> 10.0 </canopyHeight>
            <attenuationCoefficient> 1.0 </attenuationCoefficient>
        </timeSeries>
    </sensor>
</metParams>
```

Figure 11 shows velocity magnitude contour with overlaying velocity vectors of the initial velocity field created by the aforementioned example of the urban canopy profile.

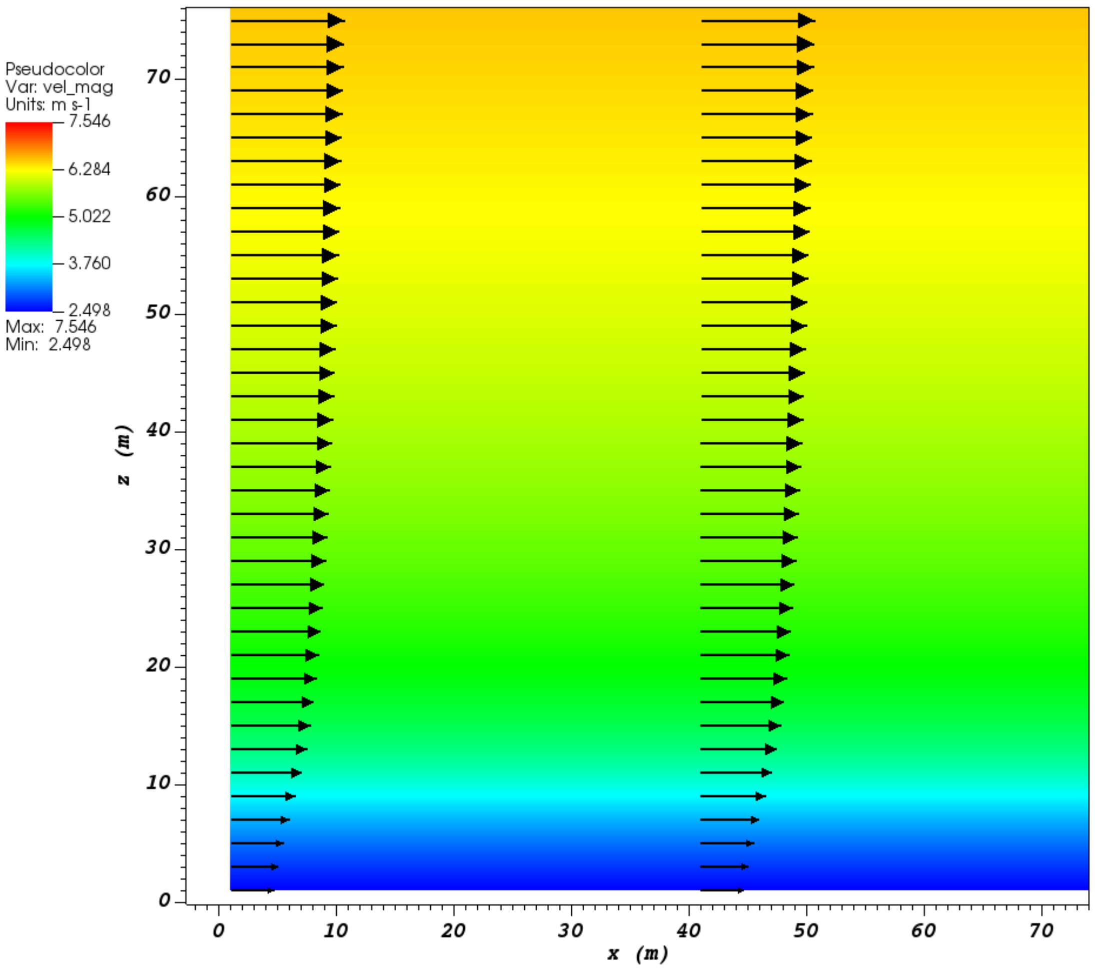


Figure 11: Velocity magnitude contour with overlaying velocity vectors in a vertical plane at $y = 101$ m for initial velocity field created by the urban canopy profile.

4) Data entry of the profile from an experimental tower with multiple sensors or from a numerical mesoscale weather prediction model like WRF [9]:

```
<metParams>
  <sensor>
      <timeSeries>      <!-- Start of timestep informastion for a sensor -->
         <boundaryLayerFlag> 4 </boundaryLayerFlag>    <!-- Site boundary layer flag (1-log, 2-
             exp, 3-urban canopy, 4-data entry) -->
         <siteZ0> 0.1 </siteZ0>          <!-- Site z0 -->
         <reciprocal> 0.0 </reciprocal>       <!-- Reciprocal Monin-Obukhov Length (1/m) -->
         <height> 30.7015 </height>        <!-- Height of the sensor -->
         <height> 74.4169 </height>
         <height> 144.644 </height>
         <height> 197.455 </height>
         <height> 268.468 </height>
         <speed> 2.56922 </speed>        <!-- Measured speed at the sensor height -->
         <speed> 2.55532 </speed>
         <speed> 2.33319 </speed>
         <speed> 2.16058 </speed>
         <speed> 1.98843 </speed>
         <direction> 323.283 </direction>      <!-- Wind direction of sensor -->
         <direction> 327.377 </direction>
         <direction> 332.676 </direction>
         <direction> 337.649 </direction>
         <direction> 344.273 </direction>
      </timeSeries>
  </sensor>
</metParams>
```

## 6 Empirical Parameterizations

QES-Winds only conserves mass and no momentum equation is solved. As a result, the solution is a potential-flow solution (no shear effects). In order to add shear effects to our solution, empirical parameterizations are needed. These parameterizations are designed using results of experiments and computational simulations (e.g. [1, 10]). Buildings are the most important elements in urban areas. There are several parameterizations developed for different areas around the building. This section covers available parameterizations in QES-Winds along with their effects on the wind field.

### 6.1 Upwind Cavity

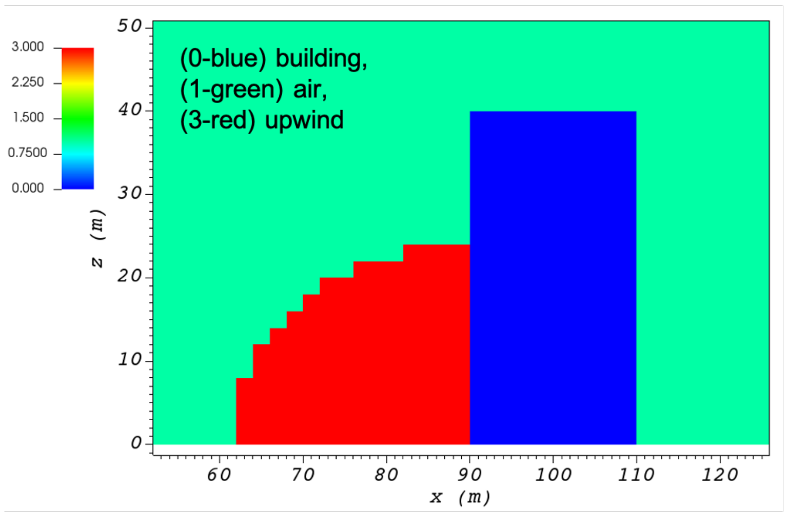
Upwind cavity as described in [11–13] is the parameterization representing upwind and stagnation effects of the building on the fluid flow. There are three options available for this type of parameterization in QES-Winds. The first option based on the parameterization proposed by Röckle [14] and later Kaplan and Dinar [15]. They defined an ellipsoid to represent what they call is the displacement zone in front of the building. The length of the displacement zone, $L_F$, is defined by Eq. 5. The shape of the ellipsoid is estimated by Eq. 6. Finally, the initial velocity components in the displacement zone are set to zero.

$$\frac{L_{\mathrm{F}}}{H} = \frac{2(W/H)}{1 + 0.8W/H} \tag{5}$$

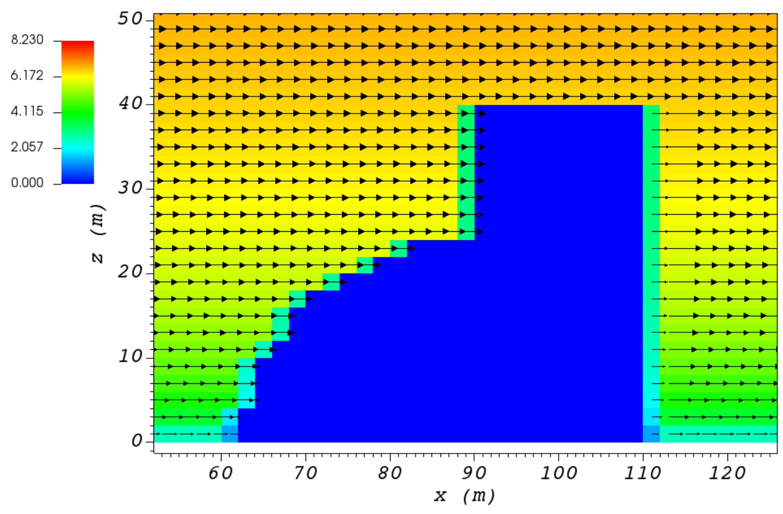
$$\frac{X^2}{L_{\mathrm{F}}^2\left(1 - (Z/0.6H)^2\right)} + \frac{Y^2}{W^2} = 1 \tag{6}$$

where $L$, $H$ and $W$ are length, width and height of the building, receptively.

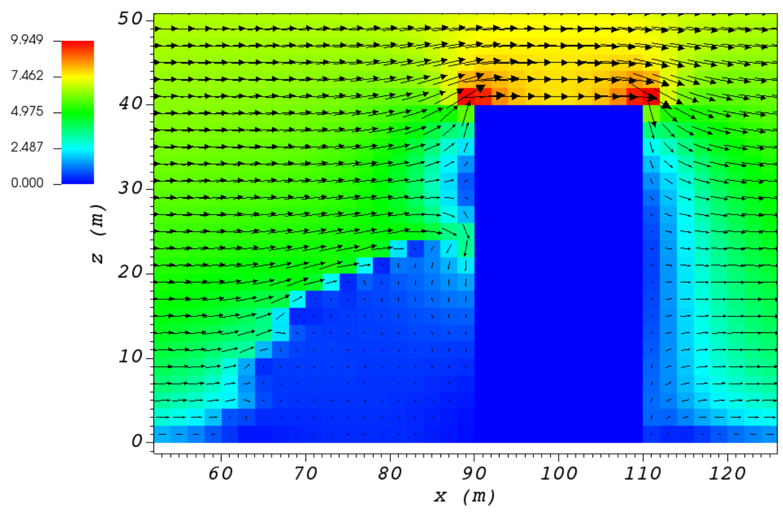
Part (a) of Figure 12 and Figure 13 show cell type contour to represent the area of effect of the Röckle upwind cavity parameterization in a vertical plane at $y$ = 100 m and a horizontal plane at $z$ = 5 m, respectively. The upwind parameterizations is applied to a rectangular building defined in Section 4.3. The initial guess field is constructed using a single sensor with logarithmic profile as defined in 5.1. Parts (b) and (c) of Figure 12 and Figure 13 indicate velocity magnitude contour with overlaying velocity vectors of initial (part (b)) and final (part(c)) velocity fields in a vertical plane at $y = 100$ m and a horizontal plane at $z = 5$ m, respectively.



(a)

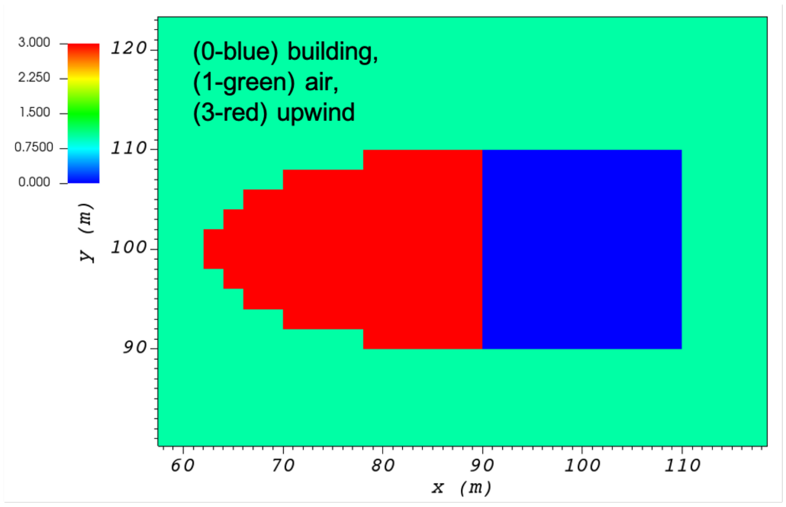


(b)

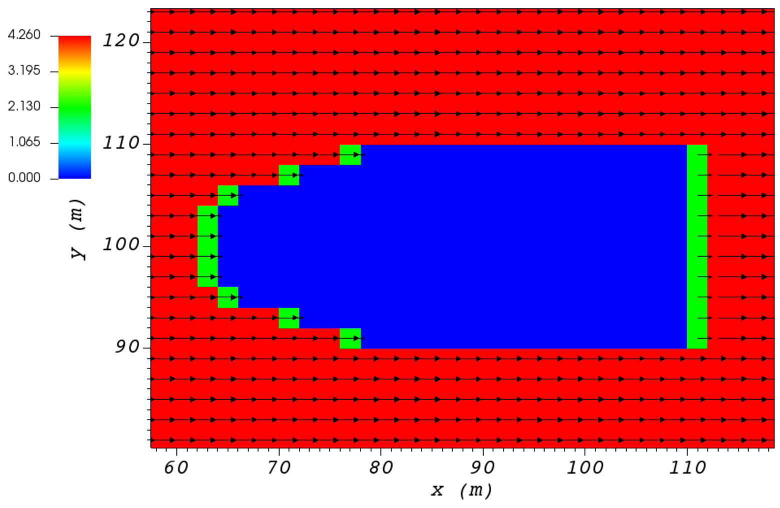


(c)

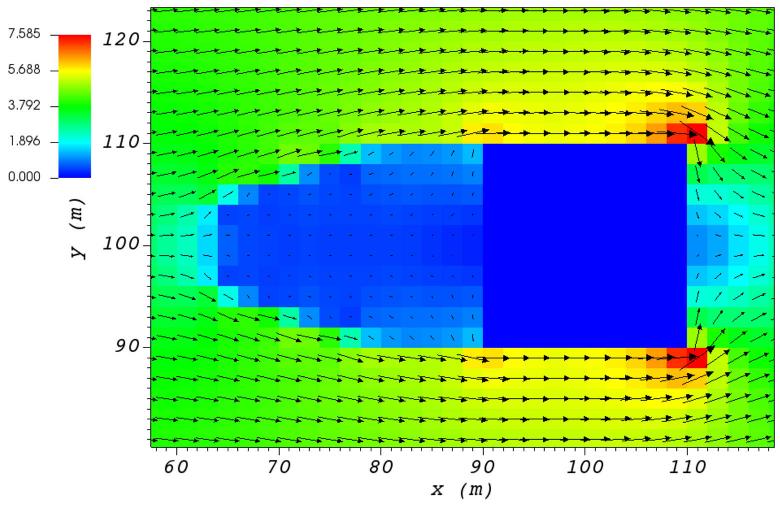
Figure 12: (a) Cell type contour to show the area of effect of the Röckle upwind cavity parameterization in a vertical plane at $y = 100$ m. Velocity magnitude contour with overlaying velocity vectors of (b) initial velocity field and (c) final velocity field, in a vertical plane at $y = 100$ m.



(a)



(b)



(c)

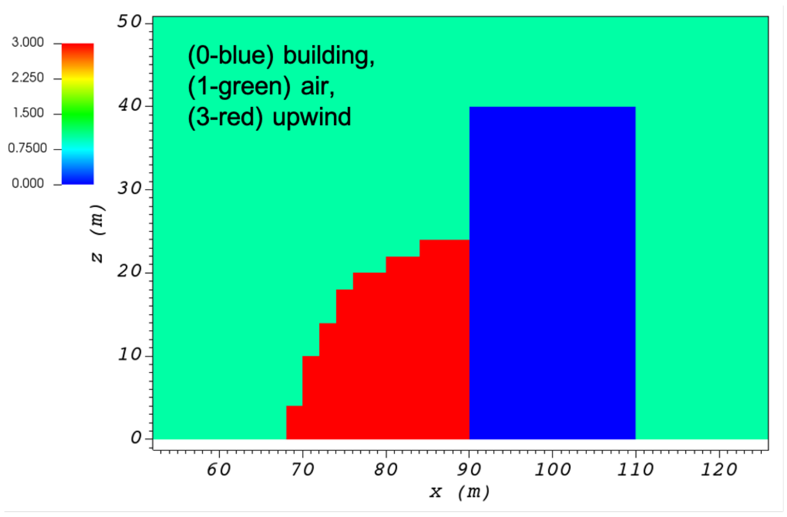
Figure 13: (a) Cell type contour to show the area of effect of the Röckle upwind cavity parameterization in a horizontal plane at $z = 5$ m. Velocity magnitude contour with overlaying velocity vectors of (b) initial velocity field and (c) final velocity field, in a horizontal plane at $z = 5$ m.

The second option is called the Modified Vortex Parameterization (MVP) and created by Bagal et al. [12]. In this parameterization, the length of the displacement zone, $L_F$, is calculated by Eq. 7. The MVP parameterization defines two ellipsoids instead of one: In the outer ellipsoid, velocities are reduced to 40% of their initial values while in the inner region, velocity components are set to zero [11]. Both ellipsoids are extended to 0.6 of the building height.

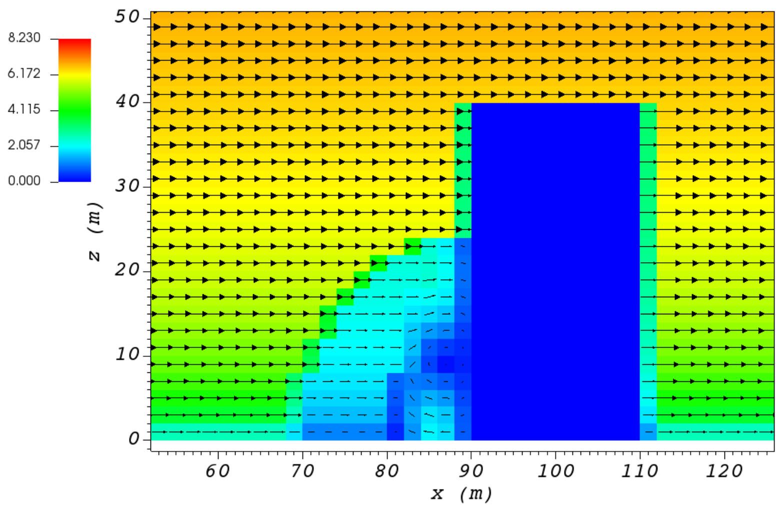
$$\frac{L_{\mathrm{F}}}{H} = \frac{1.5(W/H)}{1 + 0.8W/H} \tag{7}$$

where $L$, $H$ and $W$ are length, width and height of the building, receptively.

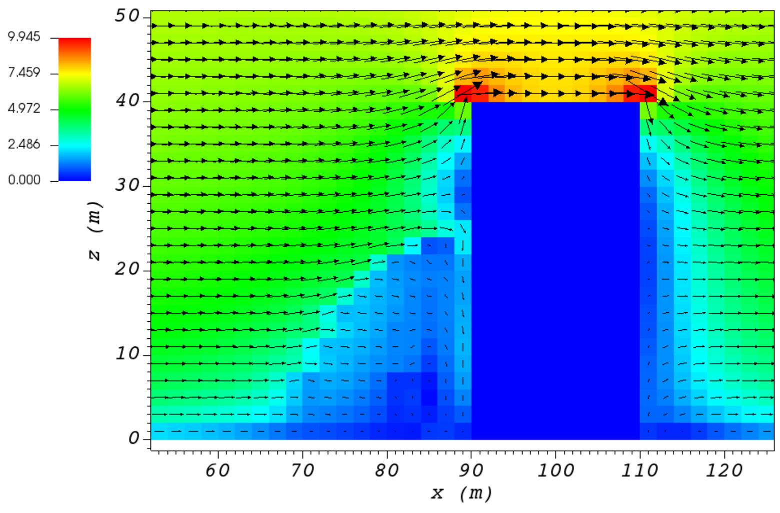
Part (a) of Figure 12 and Figure 13 show cell type contour to represent the area of effect of the MVP upwind cavity parameterization in a vertical plane at $y = 100$ m and a horizontal plane at $z = 5$ m, respectively. The upwind parameterizations is applied to a rectangular building defined in Section 4.3. The initial guess field is constructed using a single sensor with logarithmic profile as defined in 5.1. Parts (b) and (c) of Figure 12 and Figure 13 indicate velocity magnitude contour with overlaying velocity vectors of initial (part (b)) and final (part(c)) velocity fields in a vertical plane at $y = 100$ m and a horizontal plane at $z = 5$ m, respectively.



(a)

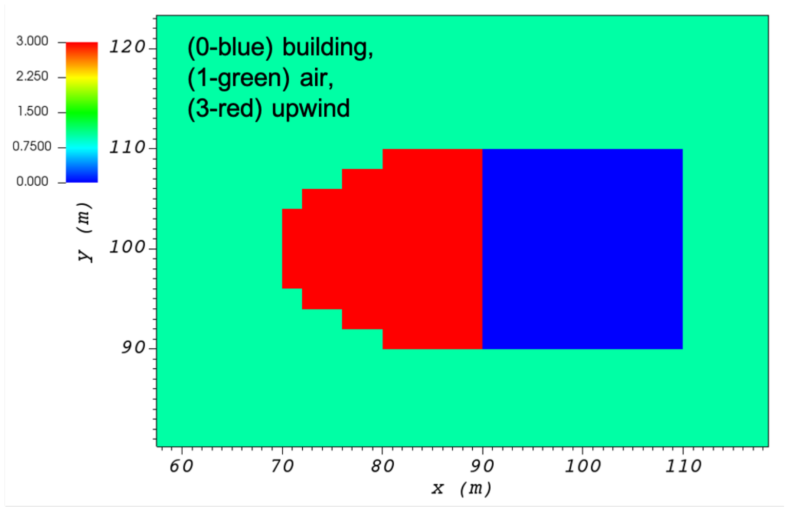


(b)

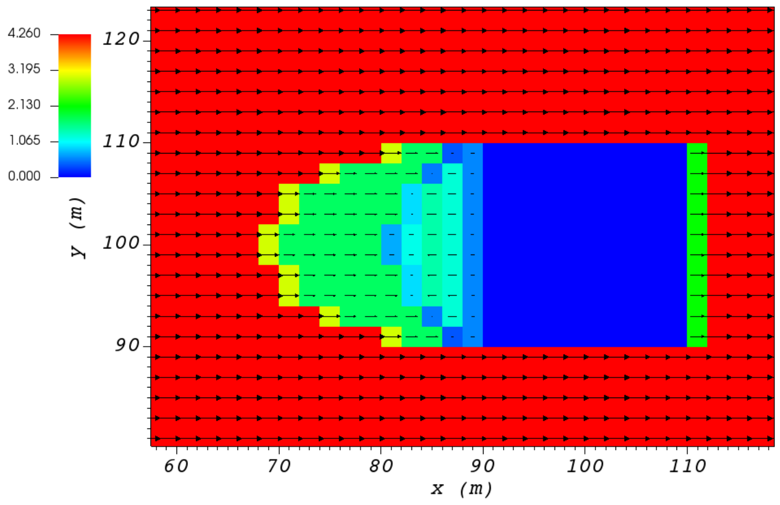


(c)

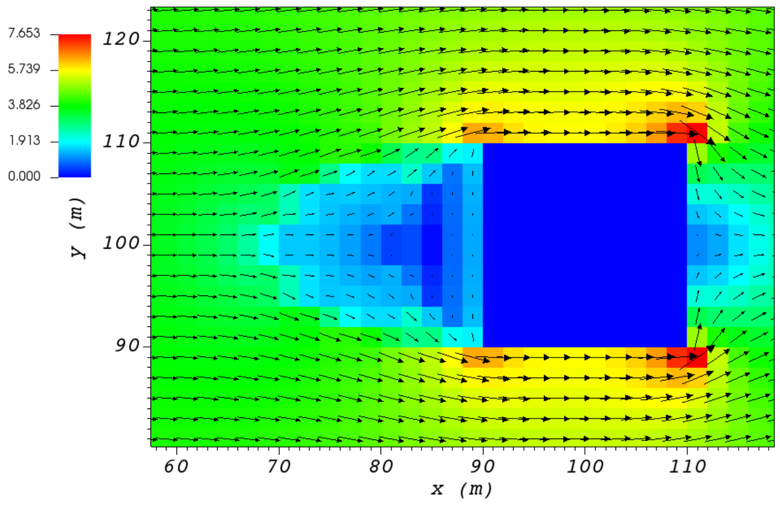
Figure 14: (a) Cell type contour to show the area of effect of the MVP upwind cavity parameterization in a vertical plane at $y = 100$ m. Velocity magnitude contour with overlaying velocity vectors of (b) initial velocity field and (c) final velocity field, in a vertical plane at $y = 100$ m.



(a)



(b)

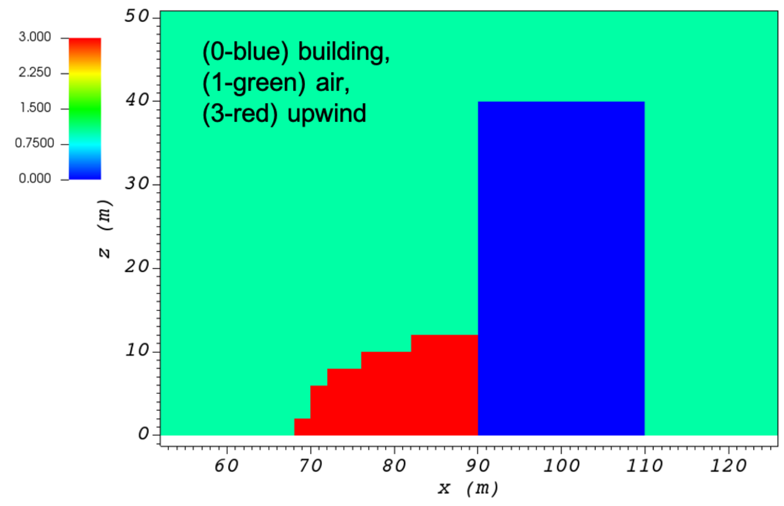


(c)

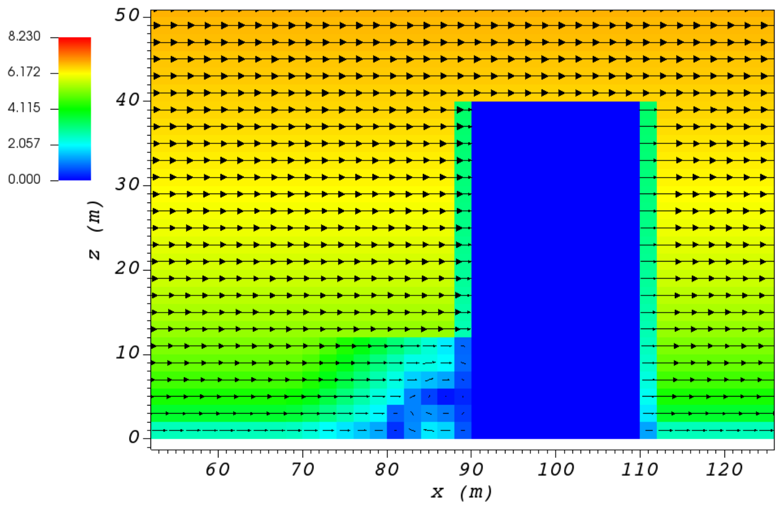
Figure 15: (a) Cell type contour to show the area of effect of the MVP upwind cavity parameterization in a horizontal plane at $z = 5$ m. Velocity magnitude contour with overlaying velocity vectors of (b) initial velocity field and (c) final velocity field, in a horizontal plane at $z = 5$ m.

The third option is called the high-rise MVP algorithm (HMVP) and is designed to address the shortcomings of the previous models when it comes to tall buildings [11]. The length of the displacement zone is calculated the same as Eq. 7. The HMVP algorithm creates two ellipsoids with the difference that the inner region only extends to 60% of the minimum of building height and building width. In addition, the algorithm linearly reduces the velocities in the outer region from their upwind values at the outer surface to 40% of the initial values on the inner region.

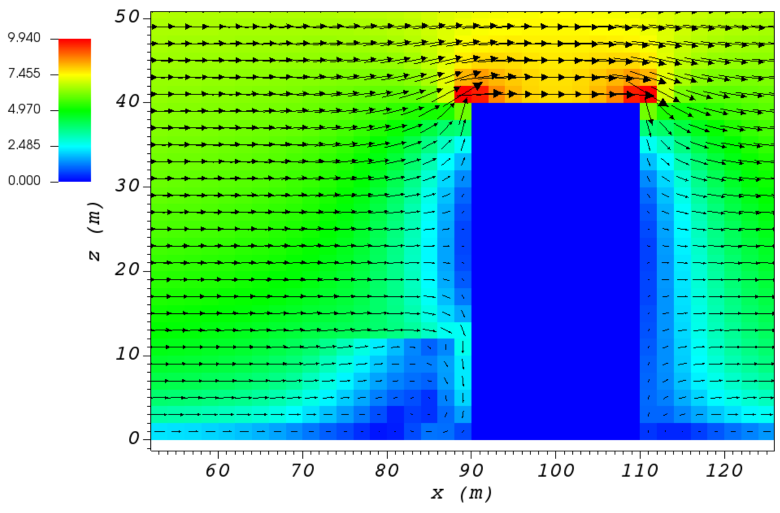
Part (a) of Figure 12 and Figure 13 show cell type contour to represent the area of effect of the HMVP upwind cavity parameterization in a vertical plane at $y = 100$ m and a horizontal plane at $z = 5$ m, respectively. The upwind parameterization is applied to a rectangular building defined in Section 4.3. The initial guess field is constructed using a single sensor with logarithmic profile as defined in 5.1. Parts (b) and (c) of Figure 12 and Figure 13 indicate velocity magnitude contour with overlaying velocity vectors of initial (part (b)) and final (part(c)) velocity fields in a vertical plane at $y = 100$ m and a horizontal plane at $z = 5$ m, respectively.



(a)

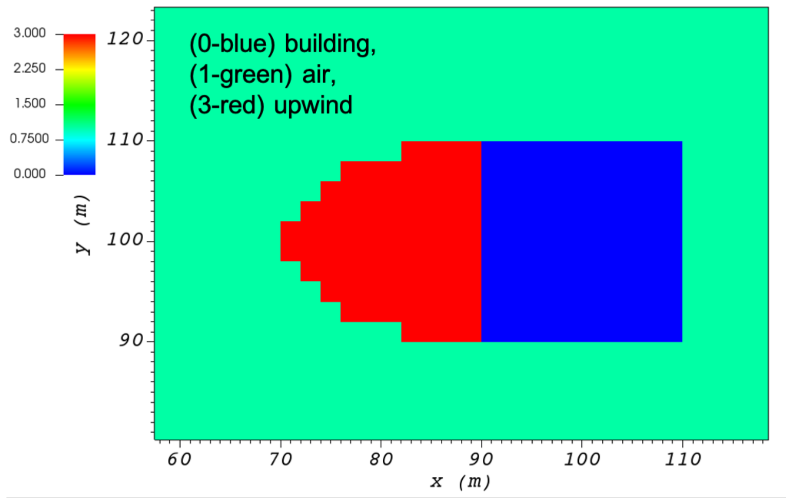


(b)

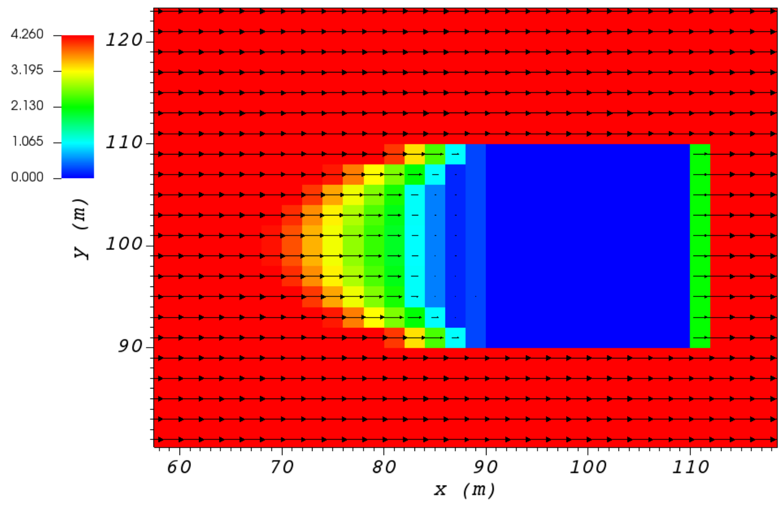


(c)

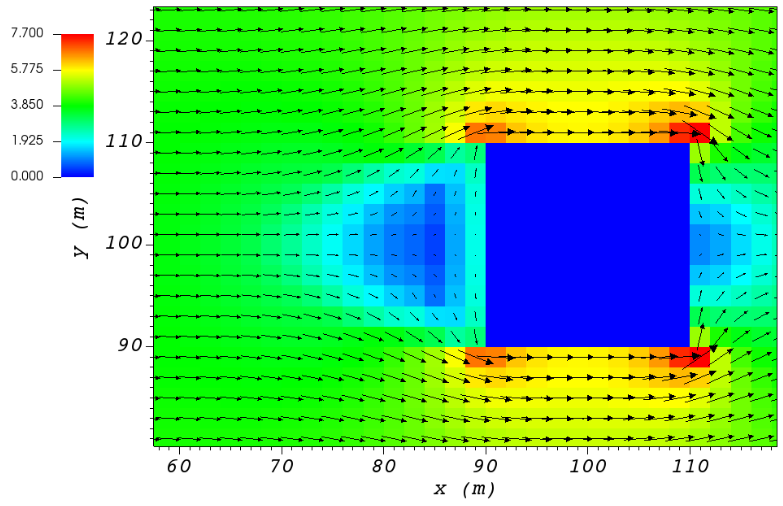
Figure 16: (a) Cell type contour to show the area of effect for the HMVP upwind cavity parameterization in a vertical plane at $y = 100$ m. Velocity magnitude contour with overlaying velocity vectors of (b) initial velocity field and (c) final velocity field, in a vertical plane at $y = 100$ m.



(a)



(b)



(c)

Figure 17: (a) Cell type contour to show the area of effect of the HMVP upwind cavity parameterization in a horizontal plane at $z = 5$ m. Velocity magnitude contour with overlaying velocity vectors of (b) initial velocity field and (c) final velocity field, in a horizontal plane at $z = 5$ m.

In order to choose between these three upwind models, the user needs to change the value of "upwindCavityFlag" in the XML file.

```
<simulationParameters>
   <upwindCavityFlag> 2 </upwindCavityFlag>    <!-- Upwind cavity flag (0-none, 1-Rockle, 2-MVP
        (default), 3-HMVP) -->
</simulationParameters>
```

## 6.2 Leeside Cavity and Far-Wake

The far-wake and cavity parameterization described in [16, 17] are a significant part of the building parameterizations. The one available in QES-Winds is based on the parameterization proposed by Röckle [14] and later Kaplan and Dinar [15]. The Röckle parameterization defines two ellipsoids to represent the shape of the reversed flow cavity and the far-wake region. The reversed flow cavity extends to the along-wind cavity length ($L_R$), which is calculated as Eq. 8, and wake is assumed to be approximately 3 cavity lengths long (i.e., $3L_R$). After calculating $L_R$, the cavity length, $d$ in the stream-wise direction was defined by an ellipsoid shape using Eq. 9. Finally, the velocity in the reversed cavity zone is defined using Eq. 10 and in the wake region, the velocity field is estimated by Eq. 11.

$$\frac{L_R}{H} = \frac{1.8\frac{W}{H}}{\left(\frac{L}{H}\right)^{0.3}\left(1 + 0.24\frac{W}{H}\right)} \tag{8}$$

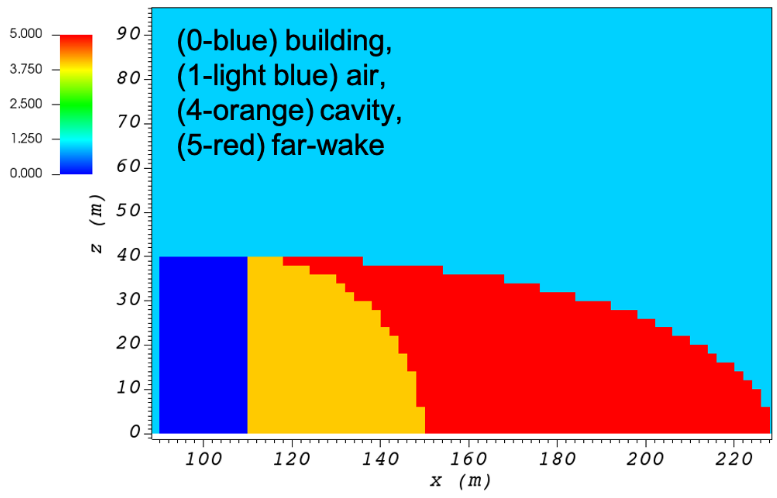
$$d = L_R\sqrt{\left(1 - \left(\frac{z}{H}\right)^2\right)\left(1 - \left(\frac{y}{W}\right)^2\right)} - \frac{L}{2} \tag{9}$$

$$\frac{u(x,y,z)}{U(H)} = -\left(1 - \left(\frac{x}{d}\right)^2\right) \tag{10}$$

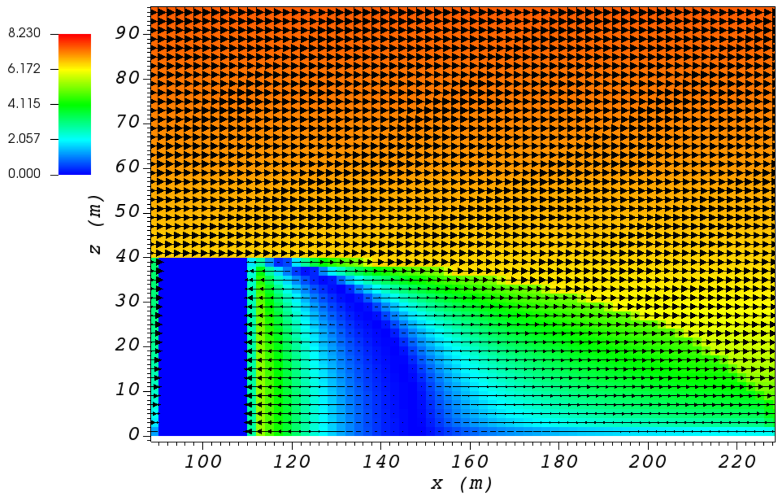
$$\frac{u(x,y,z)}{U(H)} = \left(1 - \left(\frac{d}{x}\right)^{1.5}\right) \tag{11}$$

where $L$, $H$ and $W$ are length, width and height of the building, receptively. $u(x,y,z)$ is the velocity at point $(x,y,z)$, $U(H)$ is the reference velocity at height of the building and $x$ is the distance from the building in the stream-wise direction.

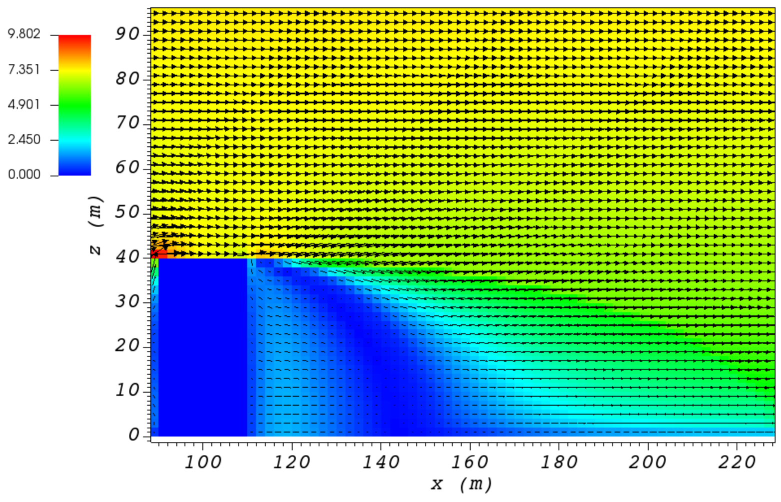
Part (a) of Figure 18 and Figure 19 show cell type contour to represent the area of effect of the Röckle wake parameterization in a vertical plane at $y = 100$ m and a horizontal plane at $z = 5$ m, respectively. The wake parameterization is applied to a rectangular building defined in Section 4.3. The initial guess field is constructed using a single sensor with logarithmic profile as defined in 5.1. Parts (b) and (c) of Figure 18 and Figure 19 indicate velocity magnitude contour with overlaying velocity vectors of initial (part (b)) and final (part(c)) velocity fields in a vertical plane at $y = 100$ m and a horizontal plane at $z = 5$ m, respectively.



(a)

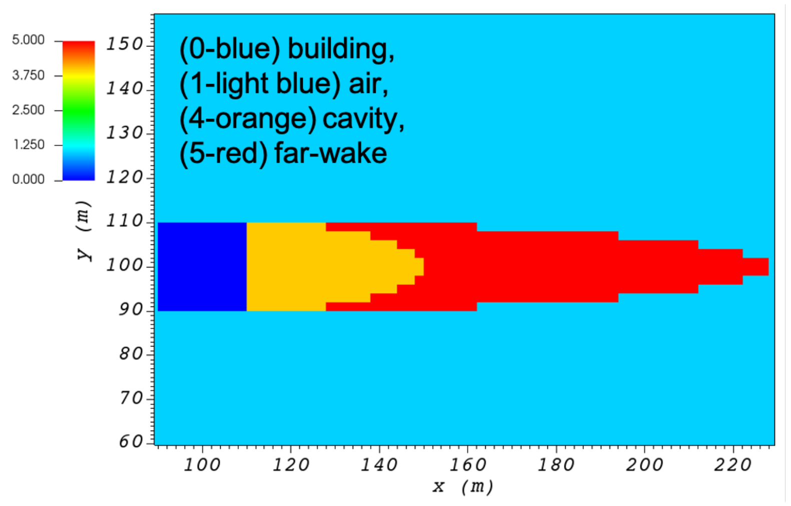


(b)

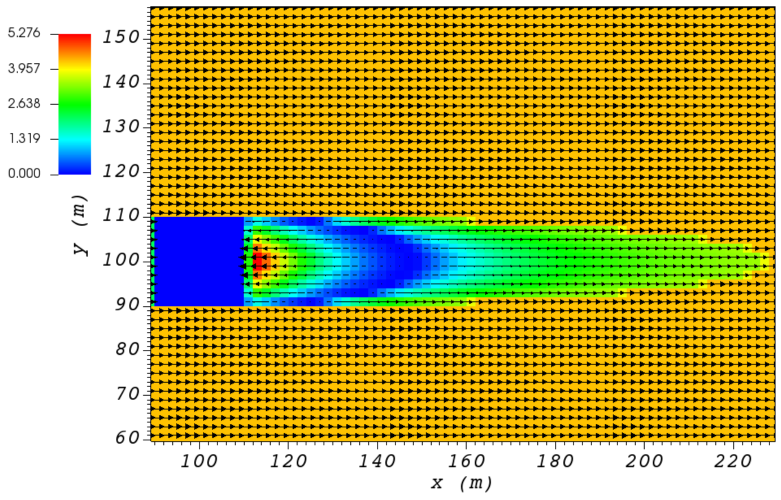


(c)

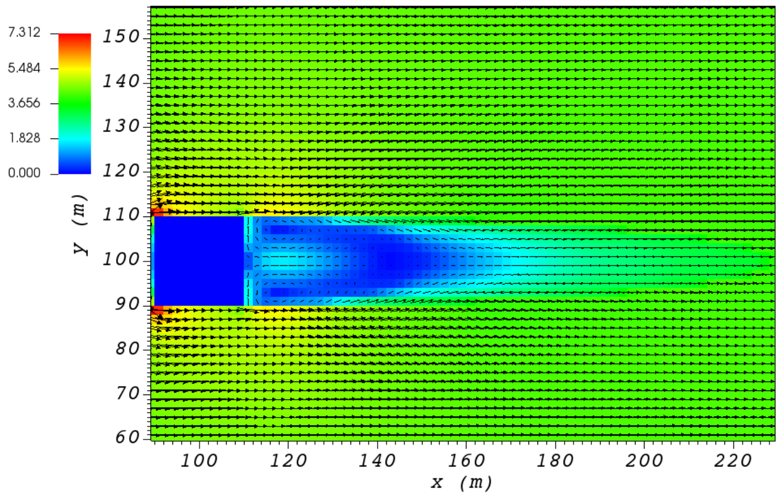
Figure 18: (a) Cell type contour to show the area of effect of the Röckle wake parameterization in a vertical plane at $y = 100$ m. Velocity magnitude contour with overlaying velocity vectors of (b) initial velocity field and (c) final velocity field, in a vertical plane at $y = 100$ m.



(a)



(b)



(c)

Figure 19: (a) Cell type contour to show the area of effect of the Röckle wake parameterization in a horizontal plane at $z = 5$ m. Velocity magnitude contour with overlaying velocity vectors of (b) initial velocity field and (c) final velocity field, in a horizontal plane at $z = 5$ m.

In order to turn on the wake model, the user needs to change the value of "wakeFlag" in the XML file.

```
<simulationParameters>
   <wakeFlag> 1 </wakeFlag>     <!-- Wake flag (0-none, 1-Rockle (default)) -->
</simulationParameters>
```

## 6.3 Street Canyon

The street canyon parameterization detailed in [10] represents the effects of two buildings in close vicinity to each other, on the fluid flow. Röckle [14] Introduced velocity parameterizations for the stream-wise components as in Eq. 12 and the vertical component as in Eq. 13.

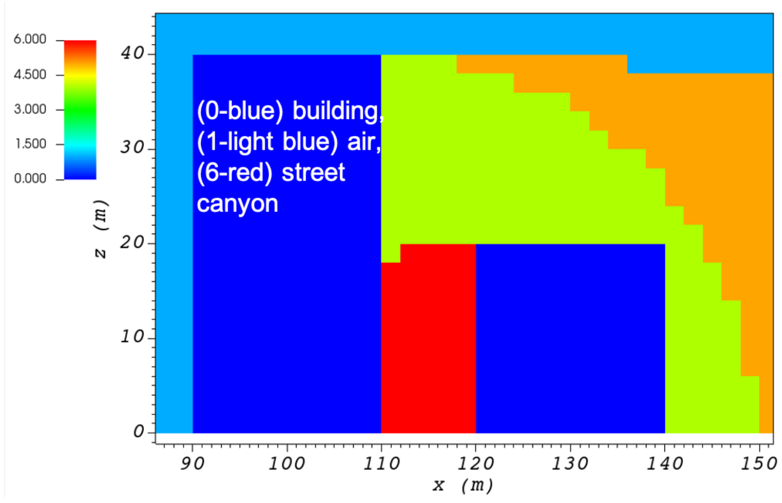
$$\frac{u(x,y,z)}{U(H)} = -\frac{x_{\text{can}}}{(0.5S)}\left(\frac{S-x_{\text{can}}}{0.5S}\right) \tag{12}$$

$$\frac{w(x,y,z)}{U(H)} = -\left|\frac{1}{2}\left(1-\frac{x_{\text{can}}}{0.5S}\right)\right|\left(1-\frac{S-x_{\text{can}}}{0.5S}\right) \tag{13}$$

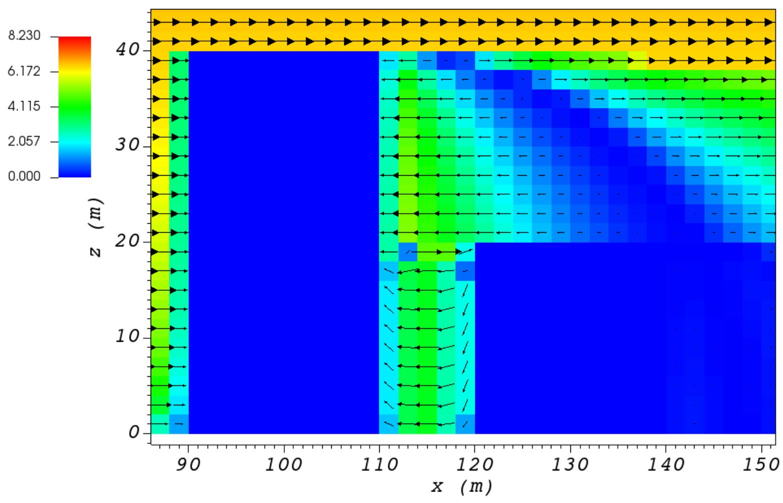
where $S$ is the spacing between two buildings and $x_{can}$ is the distance from the backwall of the upwind building.

In order to identify the criteria to determine the existence of a street canyon, Singh et al. [10] utilized the cavity length, $L_R$ (Eq. 8), for the upwind building. If $S < L_R$, the street canyon parameterization is applied, otherwise, the upwind building is considered as an isolated building.

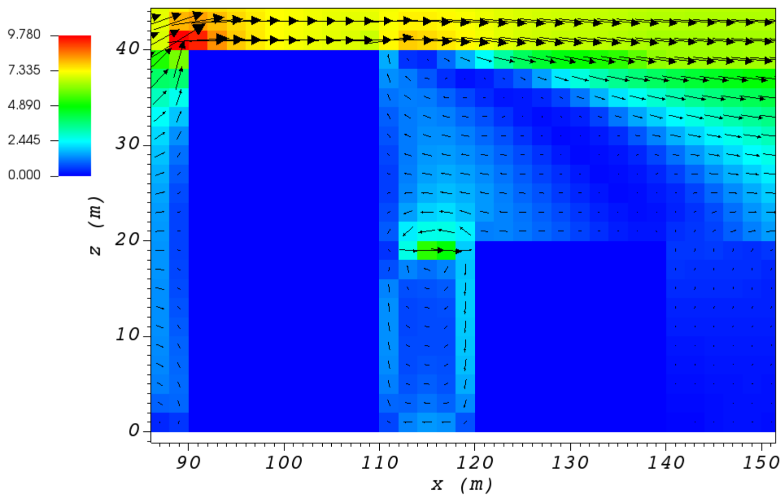
Part (a) of Figure 20 and Figure 21 show cell type contour to represent the area of effect of the street canyon parameterization in a vertical plane at $y = 100$ m and a horizontal plane at $z = 5$ m, respectively. The street canyon parameterization is applied to an area between two rectangular buildings. The upwind building is same as the one defined in Section 4.3. The downwind building is a rectangular building with 20 m as height, 0 m as base height, 20 m as length and width, closest corner to the origin located at 90 m in $x$ and 120 m in $y$ directions, and 0° as rotation angle with respect to the North-South line. The initial guess field is constructed using a single sensor with logarithmic profile as defined in 5.1. Parts (b) and (c) of Figure 20 and Figure 21 indicate velocity magnitude contour with overlaying velocity vectors of initial (part (b)) and final (part(c)) velocity fields in a vertical plane at $y = 100$ m and a horizontal plane at $z = 5$ m, respectively.



(a)

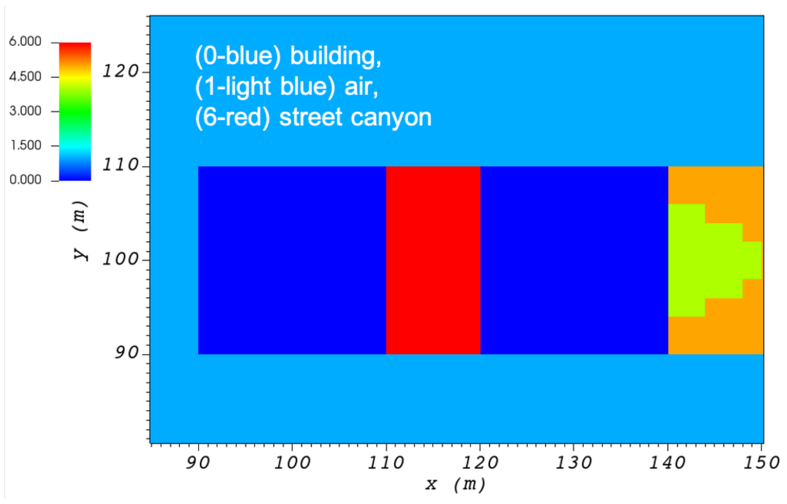


(b)

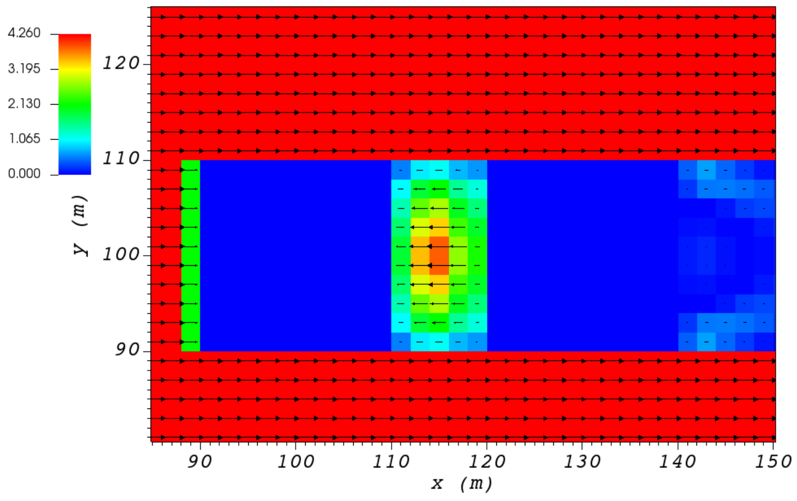


(c)

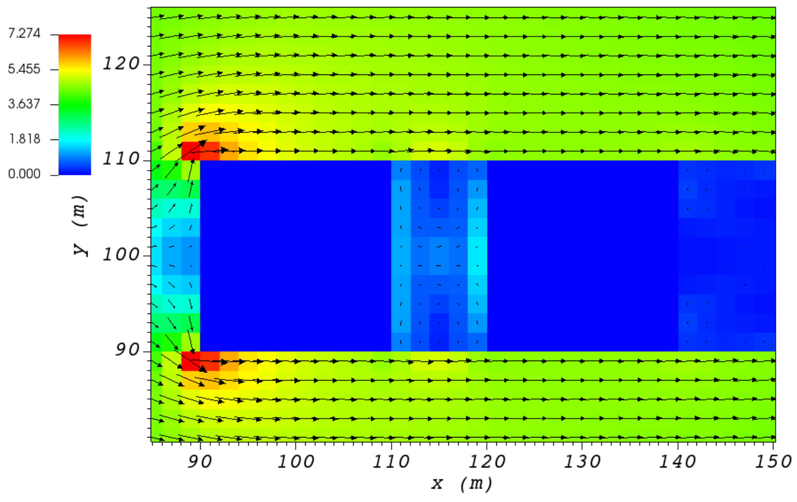
Figure 20: (a) Cell type contour to show the area of effect of the street canyon parameterization in a vertical plane at $y = 100$ m. Velocity magnitude contour with overlaying velocity vectors of (b) initial velocity field and (c) final velocity field, in a vertical plane at $y = 100$ m.



(a)



(b)



(c)

Figure 21: (a) Cell type contour to show the area of effect of the street canyon parameterization in a horizontal plane at $z = 5$ m. Velocity magnitude contour with overlaying velocity vectors of (b) initial velocity field and (c) final velocity field, in a horizontal plane at $z = 5$ m.

To turn on the street canyon parameterization, the user needs to change the value of "streetCanyon-Flag" in the XML file.

```
<simulationParameters>
   <streetCanyonFlag> 1 </streetCanyonFlag>    <!-- Street canyon flag (0-none, 1-Roeckle w/
       Fackrel (default)) -->
</simulationParameters>
```

## 6.4 Rooftop Recirculation

The rooftop parameterization described in [18, 19], captures the separation of the flow from the leading edge of the building. It first checks if the incident flow is in $\pm 15°$ of perpendicular to the front face. The parameterization then creates an ellipsoidal region above the building with height of $H_c$ (height of the vortex, calculated by Eq. 16) and length of $L_c$ (length of the vortex, calculated by Eq. 15). It applies a logarithmic profile in the whole vortex area and finally, reverses the velocity in region 1. Region 1 is an ellipsoidal zone with the same length as the vortex and half of the height.

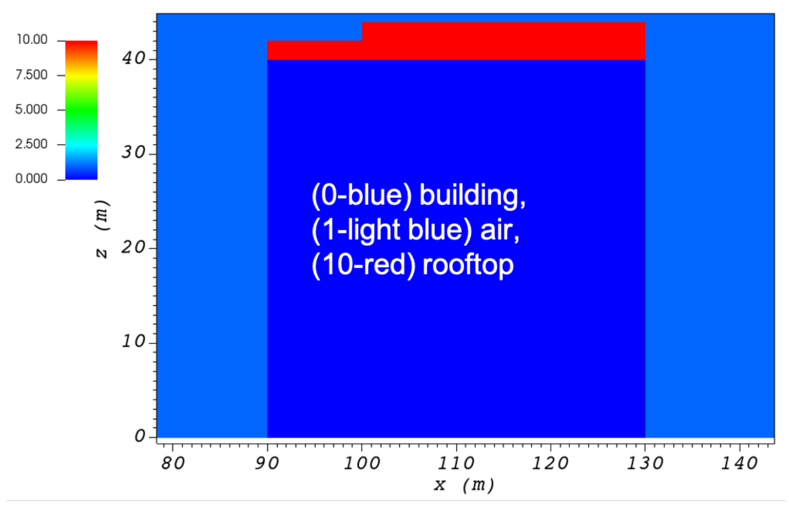
$$R = B_{\mathrm{s}}^{2/3} B_l^{1/3} \tag{14}$$

$$L_{\mathrm{c}} = 0.9R \tag{15}$$

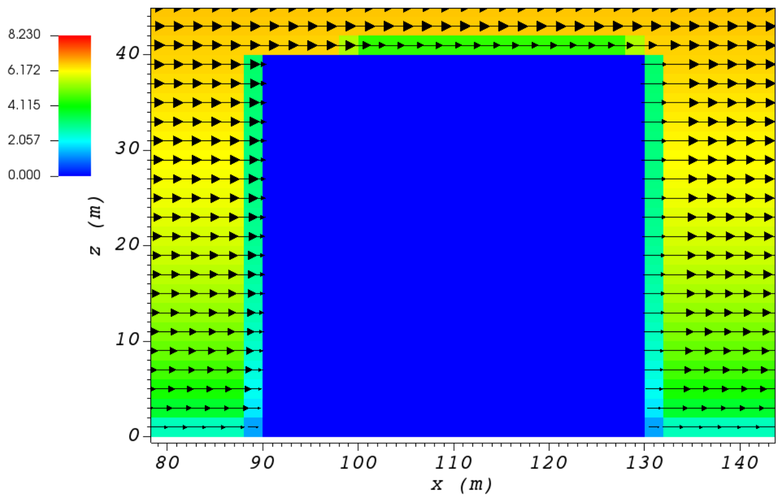
$$H_{\mathrm{c}} = 0.22R \tag{16}$$

where $B_s$ is the smaller of the height ($H$) and the effective width ($W_{eff}$) of the building, $B_l$ is the larger of $H$ and $W_{eff}$ , $R$ is the vortex size scaling factor.

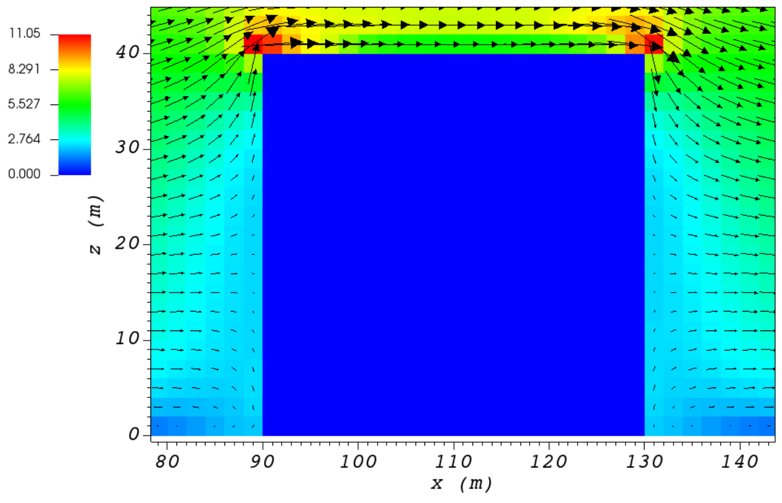
Part (a) of Figure 20 show cell type contour to represent the area of effect of the rooftop parameterization in a vertical plane at $y = 100$ m. The rooftop parameterization is applied to a rectangular building with 40 m as height, 0 m as base height, 40 m as length and width, closest corner to the origin located at 90 m in $x$ and $y$ directions, and 0° as rotation angle with respect to the North-South line. The initial guess field is constructed using a single sensor with logarithmic profile as defined in 5.1. Parts (b) and (c) of Figure 20 indicate velocity magnitude contour with overlaying velocity vectors of initial (part (b)) and final (part(c)) velocity fields in a vertical plane at $y = 100$ m.



(a)



(b)



(c)

Figure 22: (a) Cell type contour to show the area of effect of the rooftop parameterization in a vertical plane at $y = 100$ m. Velocity magnitude contour with overlaying velocity vectors of (b) initial velocity field and (c) final velocity field, in a vertical plane at $y = 100$ m.

To turn the parameterization on, the user needs to change the value of "rooftopFlag" in the XML file.

```
<simulationParameters>
   <rooftopFlag> 1 </rooftopFlag>      <!-- Rooftop flag (0-none, 1-log profile (default)) -->
</simulationParameters>
```

## 6.5 Sidewall Recirculation Zone

The sidewall parameterization is designed to represent the effects of the edge of the building on the upwind field [20]. It first checks if a face has an outward normal vector nominally ($\pm 10°$) perpendicular to the local wind vector. The important parameters controlling the sidewall vortex strength and geometry are:

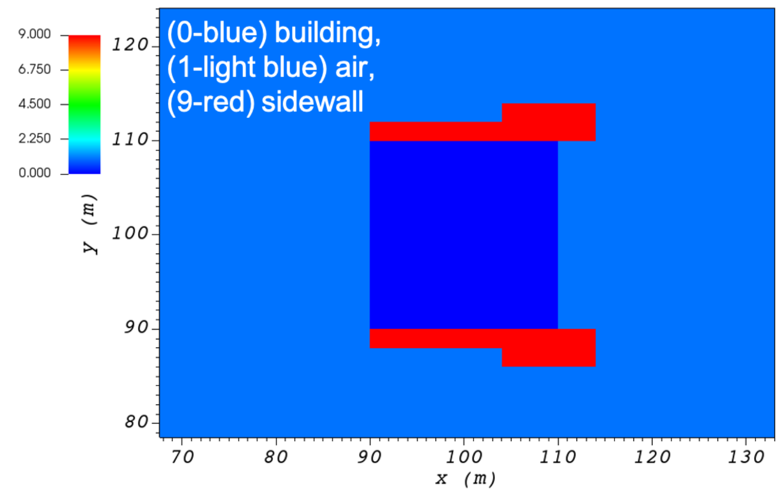
$$R = B_\mathrm{s}^{2/3} B_l^{1/3} \tag{17}$$

$$L_\mathrm{c} = 0.9R \tag{18}$$

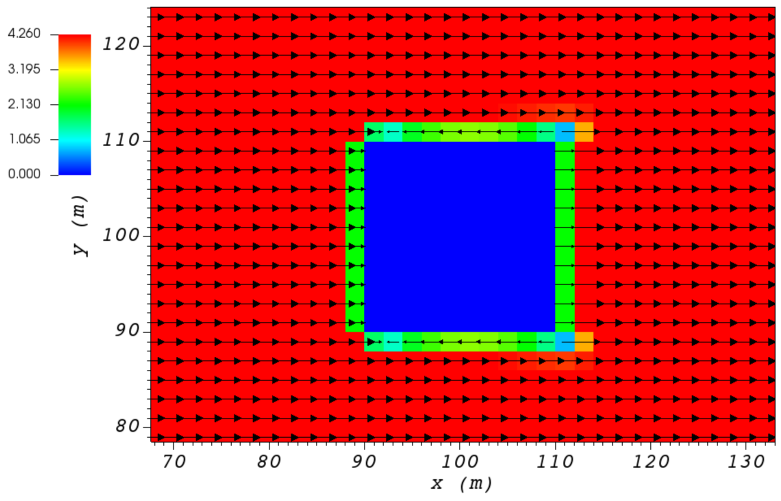
$$W_\mathrm{c} = 0.22R \tag{19}$$

where $B_s$ is the smaller of the height $(H)$ and the effective width $(W_{eff})$ of the building, $B_l$ is the larger of $H$ and $W_{eff}$ , $R$ is the vortex size scaling factor, $L_c$ is the downwind length of the half-ellipse that defines the vortex recirculation region, and $W_c$ is the lateral width of the elliptical recirculation region. Within the recirculation zone, the velocity is reversed and scaled linearly from the reference wind speed near the wall to zero at the edge of the ellipse.

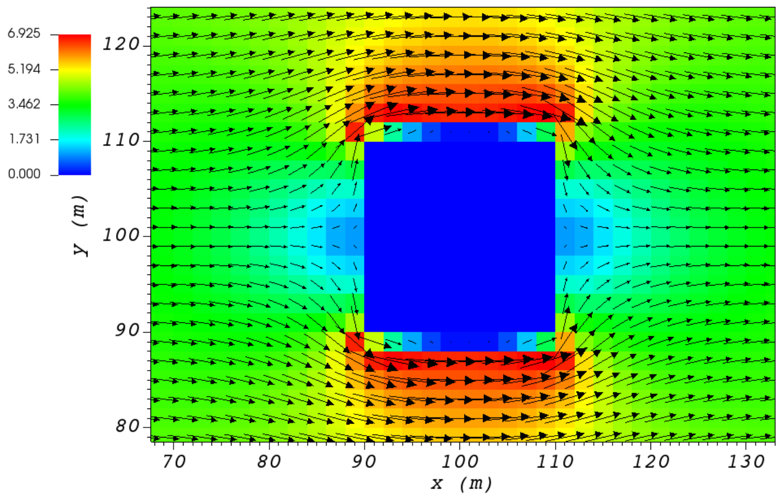
Part (a) of Figure 20 show cell type contour to represent the area of effect of the sidewall parameterization in a horizontal plane at $z = 5$ m. The rooftop parameterization is applied to a rectangular building defined in Section 4.3. The initial guess field is constructed using a single sensor with logarithmic profile as defined in 5.1. Parts (b) and (c) of Figure 23 indicate velocity magnitude contour with overlaying velocity vectors of initial (part (b)) and final (part(c)) velocity fields in a horizontal plane at $z = 5$ m.



(a)



(b)



(c)

Figure 23: (a) Cell type contour to show the area of effect of the sidewall parameterization in a horizontal plane at $z = 5$ m. Velocity magnitude contour with overlaying velocity vectors of (b) initial velocity field and (c) final velocity field, in a horizontal plane at $z = 5$ m.

In order to turn the algorithm on, the user needs to change the value of "sidewallFlag" in the XML file.

```
<simulationParameters>
   <sidewallFlag> 1 </sidewallFlag>     <!-- Sidewall flag (0-off, 1-on (default)) -->
</simulationParameters>
```

## 7 Mass Consistent Solver

QES-Winds have mass conserving wind field solvers that rapidly compute wind fields using a variational method rather than slower yet more physics based solvers that include conservation of momentum [21]. While the QES-Winds method uses reduced order physics in the numerical solution of urban flow problems, the solutions are rapid and compare quite well higher order physics-based models in both idealized [20] and realistic urban cities [22]. The method minimizes the difference between an initial wind field that is specified using empirical parameterizations [10] and the final wind fields. The empirical parameterizations account for complex wind fields around buildings such as wake cavities downstream of a building. To obtain a quasi-time-averaged velocity field, QES-Winds uses a variational analysis technique [10]. This method requires the solution of a Poisson equation for Lagrange multipliers, $\lambda$ (Equation 20) in the following form:

$$\frac{\partial^2\lambda}{\partial x^2} + \frac{\partial^2\lambda}{\partial y^2} + (\frac{\alpha_1}{\alpha_2})^2 \frac{\partial^2\lambda}{\partial z^2} = R \tag{20}$$

Where R is divergence of the initial wind field and is defined as:

$$R = -2\,\alpha_1^2 \left[ \frac{u^0_{i+1/2} - u^0_{i-1/2}}{\Delta x} + \frac{v^0_{j+1/2} - v^0_{j-1/2}}{\Delta y} + \frac{w^0_{k+1/2} - w^0_{k-1/2}}{\Delta z} \right] \tag{21}$$

The final velocity field is updated using Euler-Lagrange equations:

$$u = u^0 + \frac{1}{2\,\alpha_1^2\,\Delta x}\,[\lambda_{i+1\,,j,\,k} - \lambda_{i,\,j,\,k}] \tag{22}$$

$$v = v^0 + \frac{1}{2\,\alpha_1^2\,\Delta y}\,[\lambda_{i,\,j+1,\,k} - \lambda_{i,\,j,\,k}] \tag{23}$$

$$w = w^0 + \frac{1}{2\,\alpha_2^2\,\Delta z}\,[\lambda_{i,\,j,\,k+1} - \lambda_{i,\,j,\,k}] \tag{24}$$

The Poisson equation is solved using the Successive Over-Relaxation (SOR) method which is a variant of Gauss-Seidel method with faster convergence. Applying SOR to Equation 20 results in:

$$\lambda_{i,\,j,\,k} = \frac{\omega\left[(\Delta x)^2 R_{i,\,j,\,k} + e\,\lambda_{i+1} + f\,\lambda_{i-1} + A(g\,\lambda_{j+1} + h\,\lambda_{j-1}) + B(m\,\lambda_{k+1} + n\,\lambda_{k-1})\right]}{e+f+g+h+m+n} + (1-\omega)\lambda_{i,\,j,\,k} \tag{25}$$

Where e,f,g,h,m,n are boundary condition coefficients and A and B are domain constants. $\omega = 1.78$ is the SOR relaxation factor. The boundary condition for solid surfaces is ($\frac{\partial\lambda}{\partial n} = 0$) and for inlet/outlet surfaces it is $\lambda = 0$.

## 7.1 Solver Types

QES-Winds has four options for solving the SOR equation discussed above, the first option is to solve the equation on the CPU and the rest use the GPU for computations. The GPU solvers are called: the dynamic parallel, the global memory and the shared memory. The CPU solver is quite rapid, but slow in comparison to the GPU solvers since it is a serial solver and does not have parallel computing capabilities, especially for large domains. For more information regarding different types of solvers available in QES-Winds, read:
"Utilizing dynamic parallelism in CUDA to accelerate a 3D Red-Black Successive Over Relaxation wind-field solver"

# 8 Building and Running QES-Winds

This section is designed to serve as a step-by-step instruction of how to build and run QES-Winds. In the first part, packages required to build the code will be mentioned along with the oldest version of each package that satisfies the purpose. The next part will be interaction with the repository on GitHub in which the code is been stored to clone the code. Also, commands required for cloning the repository and building the executable of code, will be mentioned. The last part of this section will cover a brief description of how to change the input files of the code and run it.

## 8.1 Required Packages

The very first package needed to be installed is ”git” package. It provides the ability to interact with GitHub and use commands to clone the repository, switch between different branches and etc. This package does not have any dependencies, so it is always recommended to install the latest version. The next package inline is ”CMake” and its GUI version ”CCMake”. It finds all the packages required, links them together and creates the ”makefile” for building the code. CMake should be any version greater than 3.10. QES-Winds also needs ”boost” libraries in order to have access to C++ source libraries. Boost 1.66.0 is sufficient for the purpose of QES-Winds. ”Gdal” libraries are necessary to read in Digital Elevation Models (DEM) and shapefile (for buildings). Version 2.3.1 of gdal libraries will do the job for our applications. The last library that needs to be installed is ”netcdf-c” libraries along with netcdf interface with C++, version 4.6 is required. Netcdf libraries are essential for reading in WRF output files and writing QES-Winds results in netcdf format. Finally, since QES-Winds is written in C++ and CUDA, ”gcc” and ”CUDA” compilers needed to be installed. Because there is a compatibility issue between versions of CUDA, gcc and Operating System(OS) (for more information go to https://docs.nvidia.com/cuda/cuda-installation-guide-linux/index.html), version of gcc that is compatible with the version of CUDA and OS is required. For CUDA, at least version 8.0 needs to be installed.

## 8.2 Cloning QES-Winds from GitHub

After making sure all the required packages are installed and ready to use, a copy of QES-Winds needs to be downloaded on the local computer (cloning process). To clone the code, go to the directory you want to have the code downloaded, open a terminal and type ”git clone [address to the repository]”. To get the address to the repository, go to the repository GitHub page, UtahEFD/QES-Winds-Public, click on the green button ”Code” and copy the ”HTTPS” address. It downloads a copy of the code in the âĂIJmasterâĂİ branch of the repository in your local directory.

## 8.3 Building Executable of QES-Winds

Next steps are:

- Go to the folder created with name QES-Winds: ”cd QES-Winds-Public”.
- Create a build directory: ”mkdir build” or ”sudo mkdir build”.
- Go to folder build: ”cd build”.
- Type: ”cmake ..”.

There is a chance that cmake fails to find all the packages needed for running the code (packages installed on unconventional directories). In this case, you need to do cmake with appropriate flags that point to those packages.

- After cmake is done successfully, type: ”build”
- A successful build will result in creating the executable named ”qesWinds”

## 8.4 Running QES-Winds

The command to run the QES-Winds executable created above is:

```
./qesWinds/qesWinds -q [address to XML file] -o [output file] -s [solver type] -z [Visualization output]
```

At least three elements need to be addressed: input XML file, output file name and type of solver. The input XML file defines various variables necessary for running the code. Input files are usually located in "QES-Winds/data/InputFiles" and defined in command line by "-q".

```
[address to XML file] = QES-Winds/data/InputFiles/XMLfilename
```

User can change the name of output file by "-o" outputname. QES-Winds has four solver types: solving on CPU (determined by "-s 1"), solving SOR equation on GPU using dynamic parallelism (determined by "-s 2"), GPU solver using global memory (determined by "-s 3") and GPU solver using shared memory (determined by "-s 4"). GPU solvers are much faster than CPU solver and are highly recommended especially for large domains.

# Bibliography

[1] M. J. Brown, A. A. Gowardhan, M. A. Nelson, M. D. Williams, and E. R. Pardyjak, "Quic transport and dispersion modelling of two releases from the joint urban 2003 field experiment," *International Journal of Environment and Pollution*, vol. 52, no. 3-4, pp. 263–287, 2013.

[2] P. Taylor and H. Teunissen, "The askervein hill project: overview and background data," *Boundary-layer meteorology*, vol. 39, no. 1-2, pp. 15–39, 1987.

[3] R. Mickle, N. Cook, A. Hoff, N. Jensen, J. Salmon, P. Taylor, G. Tetzlaff, and H. Teunissen, "The askervein hill project: Vertical profiles of wind and turbulence," *Boundary-Layer Meteorology*, vol. 43, no. 1-2, pp. 143–169, 1988.

[4] K. J. Allwine and J. E. Flaherty, "Joint urban 2003: Study overview and instrument locations," tech. rep., Pacific Northwest National Lab.(PNNL), Richland, WA (United States), 2006.

[5] T. Favaloro, E. R. Pardyjak, and M. J. Brown, *Toward Understanding the Sensitivity of the QUIC Dispersion Modeling System to Real Input Data.* PhD thesis, Department of Mechanical Engineering, University of Utah, 2008.

[6] E. Pardyjak, S. Speckart, F. Yin, and J. Veranth, "Near source deposition of vehicle generated fugitive dust on vegetation and buildings: Model development and theory," *Atmospheric Environment*, vol. 42, no. 26, pp. 6442–6452, 2008.

[7] S. E. Koch, M. DesJardins, and P. J. Kocin, "An interactive barnes objective map analysis scheme for use with satellite and conventional data," *Journal of climate and applied meteorology*, vol. 22, no. 9, pp. 1487–1503, 1983.

[8] T. M. Booth and E. Pardyjak, *Validation of a data assimilation technique for an urban wind model.* Department of Mechanical Engineering, University of Utah, 2012.

[9] J. G. Powers, J. B. Klemp, W. C. Skamarock, C. A. Davis, J. Dudhia, D. O. Gill, J. L. Coen, D. J. Gochis, R. Ahmadov, S. E. Peckham, *et al.*, "The weather research and forecasting model: Overview, system efforts, and future directions," *Bulletin of the American Meteorological Society*, vol. 98, no. 8, pp. 1717–1737, 2017.

[10] B. Singh, B. S. Hansen, M. J. Brown, and E. R. Pardyjak, "Evaluation of the quic-urb fast response urban wind model for a cubical building array and wide building street canyon," *Environmental fluid mechanics*, vol. 8, no. 4, pp. 281–312, 2008.

[11] M. Nelson, B. Addepalli, F. Hornsby, A. Gowardhan, E. Pardyjak, and M. Brown, "5.2 improvements to a fast-response urban wind model," 2008.

[12] N. Bagal, E. Pardyjak, and M. Brown, "Improved upwind cavity parameterization for a fast response urban wind model," in *84th Annual AMS Meeting. Seattle, WA*, 2004.

[13] A. A. Gowardhan, M. J. Brown, and E. R. Pardyjak, "Evaluation of a fast response pressure solver for flow around an isolated cube," *Environmental fluid mechanics*, vol. 10, no. 3, pp. 311–328, 2010.

[14] R. Röckle, *Bestimmung der Strömungsverhältnisse im Bereich komplexer Bebauungsstrukturen.* na, 1990.

[15] H. Kaplan and N. Dinar, "A lagrangian dispersion model for calculating concentration distribution within a built-up domain," *Atmospheric Environment*, vol. 30, no. 24, pp. 4197–4207, 1996.

[16] B. Singh, *Testing and development of a fast response Lagrangian dispersion models.* PhD thesis, Department of Mechanical Engineering, University of Utah, 2005.

[17] B. Singh, E. Pardyjak, M. Brown, and M. Williams, "Testing of a far-wake parameterization for a fast response urban wind model," in *Sixth Symposium on the Urban Environment/14th Joint Conference on the Applications of Air Pollution Meteorology with the Air and Waste Management Association, Atlanta, January J*, vol. 8, 2006.

[18] N. Bagal, B. Singh, E. R. Pardyjak, and M. J. Brown, "Implementation of rooftop reciculation parameterization into the quic fast response urban wind model," tech. rep., Los Alamos National Laboratory, 2004.

[19] S. Pol, N. Bagal, B. Singh, M. Brown, and E. Pardyjak, "Implementation of a rooftop recirculation parameterization into the quic fast response urban wind model," 2006.

[20] A. N. Hayati, R. Stoll, J. Kim, T. Harman, M. A. Nelson, M. J. Brown, and E. R. Pardyjak, "Comprehensive evaluation of fast-response, reynolds-averaged navier–stokes, and large-eddy simulation methods against high-spatial-resolution wind-tunnel data in step-down street canyons," *Boundary-Layer Meteorology*, vol. 164, no. 2, pp. 217–247, 2017.

[21] J.-J. Kim, E. Pardyjak, D.-Y. Kim, K.-S. Han, and B.-H. Kwon, "Effects of building-roof cooling on flow and air temperature in urban street canyons," *Asia-Pacific Journal of Atmospheric Sciences*, vol. 50, no. 3, pp. 365–375, 2014.

[22] M. Neophytou, A. Gowardhan, and M. Brown, "An inter-comparison of three urban wind models using oklahoma city joint urban 2003 wind field measurements," *Journal of Wind Engineering and Industrial Aerodynamics*, vol. 99, no. 4, pp. 357–368, 2011.