COMPACT User Manual

Using the **Co**mpact **M**odel of **P**otential flow **a**nd **C**onvective **T**ransport

Version 2.0

1/26/2012
UC Berkeley Department of Mechanical Engineering
Michael Toulouse

Table of Contents

Introduction

This is the user's manual to COMPACT, short for Compact Model of Potential Flow and Convective Transport. It is intended as a first-order tool for use in data center design and organization. It is not intended to supplant CFD entirely, but more to narrow the design parameter space down before more time-consuming models are used. Other uses include adaptation for real-time controllers, and assessment of data center operational exergy destruction in life-cycle exergy assessment.

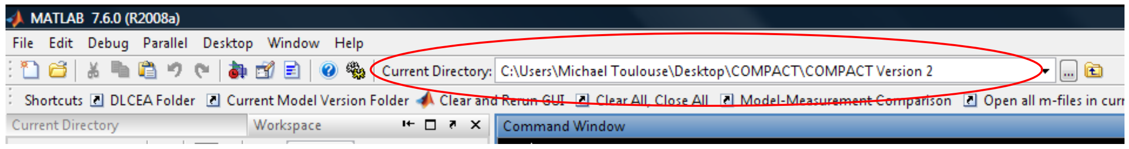
Note that this code has two iterations: The first, version 1.0, involved a potential flow solver and temperature solver with no corrections. Measurements of a real data center showed an excessively high (by 20+ degrees C) prediction of temperature by the model in specific areas, and in response version 2.0 included code to create a flow field correction based on a vortex superposition method (described later). This manual is for version 2.0.

Setting up MATLAB to run the model

The first step in running the model is to copy over the folder containing all the relevant m-files. The name maybe "Compact Flow + Temp Model Rebuild" or some other name, but it does not matter so long as the file contains several folders with names like "@Room" and "@Inlet" and , and some individual files like "makeroom.m".

Once in MATLAB, set the current directory through one of two ways:

1) Manually change the current directory through the pop-up menu located near the top of the MATLAB window.


MATLAB 7.6.0 (R2008a)
File Edit Debug Parallel Desktop Window Help
Shortcuts DLCEA Folder Current Model Version Folder Clear and Rerun GUI Clear All, Close All Model-Measurement Comparison Open all m-files in curr
Current Directory Workspace Command Window

Current Directory: C:\Users\Michael Toulouse\Desktop\COMPACT\COMPACT Version 2

2) Change the current directory from the command line, by using the "cd" command

```
>> cd('C:\Users\Michael Toulouse\Desktop\Compact Version 2')
```

MATLAB is now set up to run the model.

Starting the GUI

Once MATLAB has been set up, the model GUI is started by the "makeroom" function. The output for the makeroom function is a Room-class object, so the room variable can be assigned a name in the MATLAB base workspace:

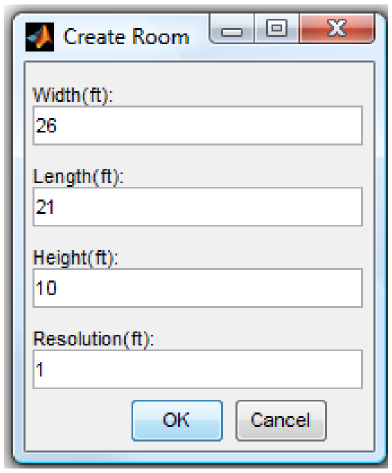
```
>> makeroom

>> Rm = makeroom;

>> makeroom;
```

All the above are legitimate commands. The semicolon suppresses the display of information giving information about the Room object, which would look something like:

```
26x21x10 meter room with:
0 Racks
0 Inlets
0 Outlets
0 Partitions
0 Obstacles
Resolution = 1 feet
```

Similarly, all class display methods have been defined, so entering any variable name for an object in a customized class will yield information about the object. However, the variable information will not display at all unless the makeroom function is completed.

After "makeroom" is called, an input dialog will pop up prompting the user for the room dimensions and resolution:


Create Room

Width(ft): 26

Length(ft): 21

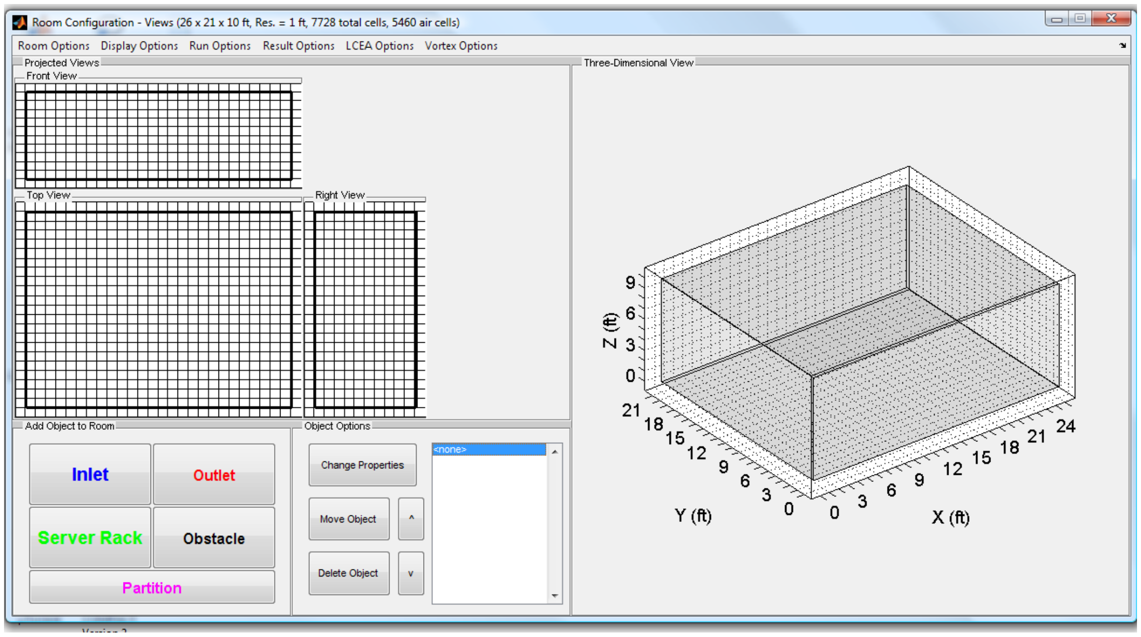
Height(ft): 10

Resolution(ft): 1

Once 'OK' is pressed, the function "room_shape" is called, which creates the main GUI.

Main GUI

The main GUI will look like the screenshot below, though there may be differences due to varying room dimensions. The program has been written to adapt the views to any legitimate room dimensions.



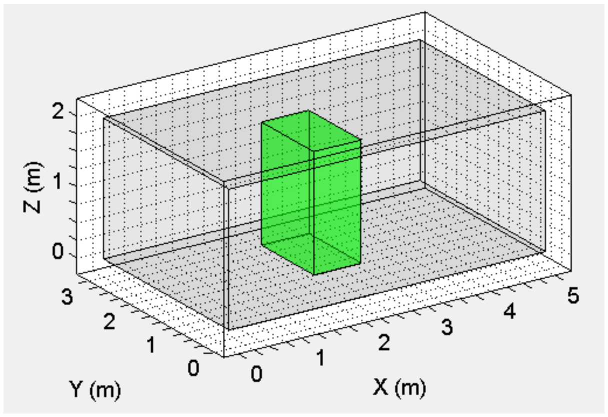
There are several relevant parts of this GUI, each of which is explained in more detail.

- The room views
- The menu options
- The object addition buttons
- The object edit/delete buttons

Room Views

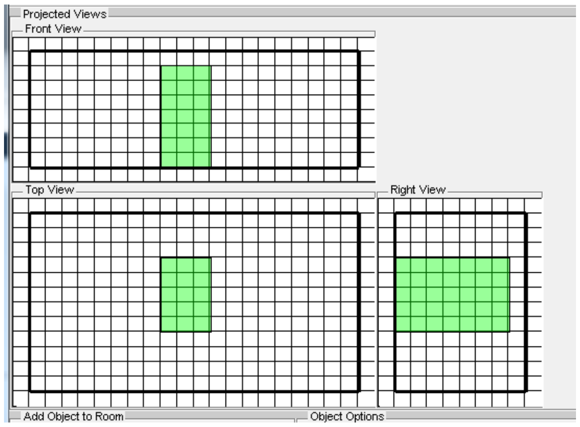
There are two room views: 3D View and Projected View. In the explanations below, a server rack has been placed for demonstration purposes.

3D

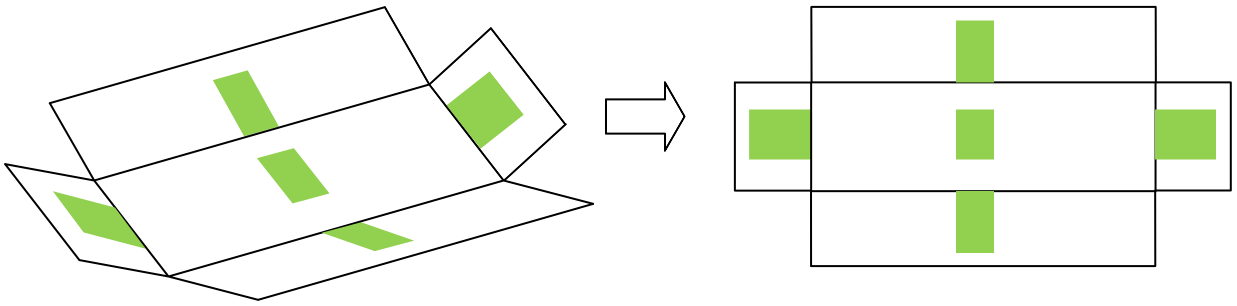


The 3D view shows a three-dimensional representation of the room. Note the labeling of the axes. The gray box represents the walls, floor, and ceiling of the room, with one extra resolution unit past each wall. This was originally to allow for display of inlets and outlets in version 1.0, but version 2.0 now uses zero-depth "patches" to represent inlets and outlets instead of boxes which extended into the wall. The view can be rotated with the mouse to any angle.

Projected



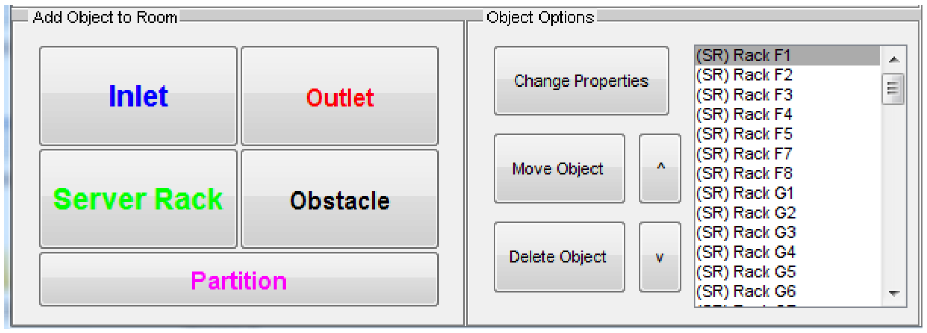
The projected views give the 3D view as projected onto given planes. The best way to think of the projected views is to think of the room as a foldable box. Each object within is projected onto each wall, then the room is unfolded flat:



The middle, top and right walls of the box are the projected views seen in the main GUI. The bottom and left projections are not necessary to fully understand the position of an object, and are omitted. However, they are useful when placing outlets on the bottom or left walls.

Buttons/Object List

Below the projected views are the buttons and lists controlling the selection, adding, editing, and deleting of flow objects in the room. Addition buttons are leftmost, followed by edit/delete buttons and the object list.

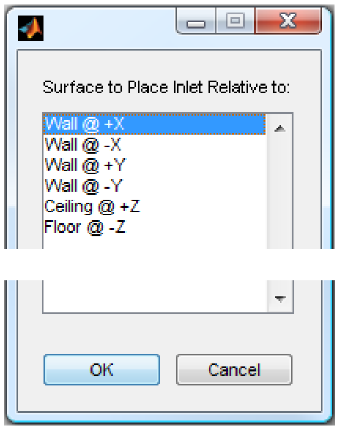


The graphical representation of each of these objects in the 3D view is roughly the same color as the button's text (server racks are green, inlets are blue, etc.).

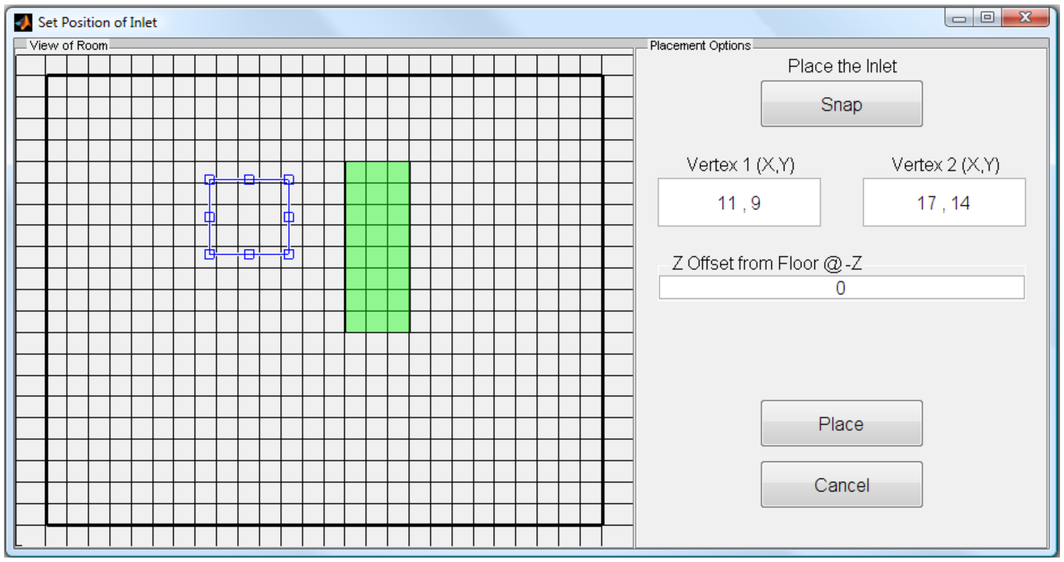
Adding Objects

There are five different types of flow objects you can add: **server racks, inlets, outlets, obstacles, and partitions**. The first three are fairly self-explanatory, but to clarify, obstacles are 3-D objects placed in the room, essentially the same as having a server rack with zero heat generation and flow rate. Partitions are zero-thickness barriers that can be placed anywhere in the room and will prevent flow across them.

When one of the Add Object buttons is pressed, a GUI opens to allow placement of the object. First, it asks which surface of the room you want to use as a reference plane.



Note that this step is skipped if server rack is chosen, as the default reference plane for server rack is the floor. Next is the main position-setting GUI.



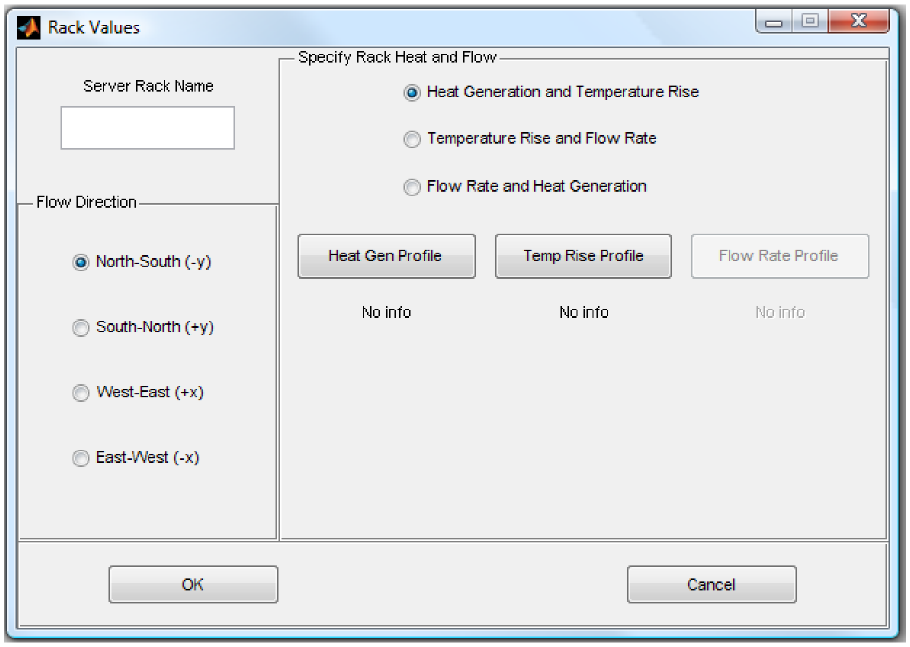
Vertex 1 (X,Y): 11 , 9

Vertex 2 (X,Y): 17 , 14

Z Offset from Floor @ -Z: 0

In the position-setting GUI the appropriate projected view is shown, and a rectangle is placed within the room view, which can be moved or dragged to the appropriate size and shape. The “Snap” button snaps the rectangle to the resolution grid, and “Place” sets the position of the object and exits the position GUI. There’s also the option of “offset”, which places your object some distance (closer to you) from the reference plane. If a server rack is being placed, the GUI has an extra field for rack height, which is also subject to the “Snap” function.

Once the position of the object is defined, one of two GUIs is called, this time to set the flow properties of the object:



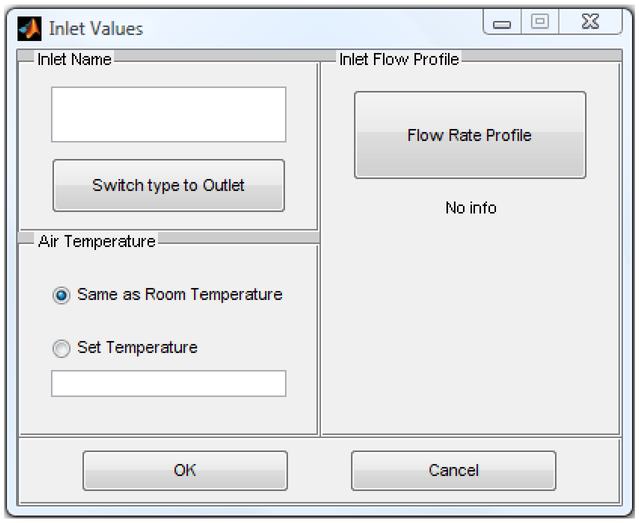
[✓ Heat Generation and Temperature Rise] Temperature Rise and Flow Rate Flow Rate and Heat Generation

Flow Direction: [✓ North-South (-y)] South-North (+y) West-East (+x) East-West (-x)

Heat Gen Profile: No info

Temp Rise Profile: No info

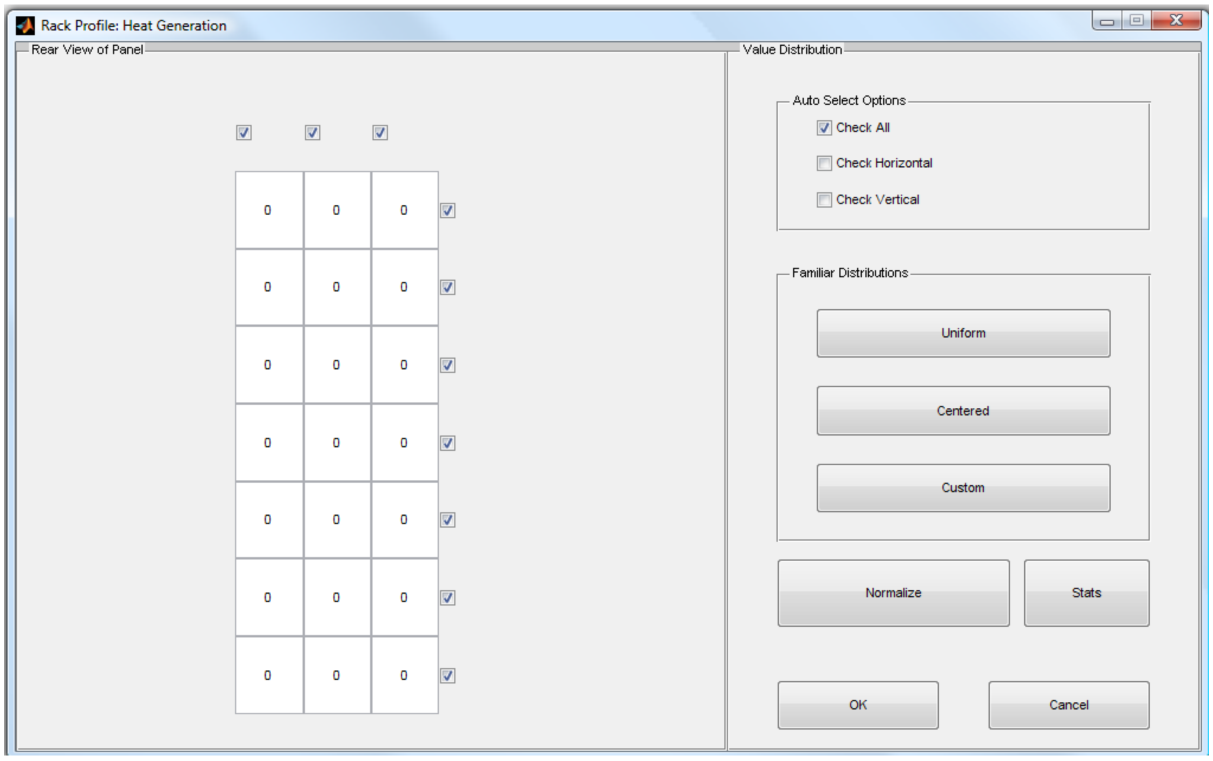
Flow Rate Profile: No info



The inlet/outlet property editor GUIs (they are essentially the same, with different text) simply ask for the name (optional), the air temperature and the flow profile, but the server rack property editor also includes flow direction and the heat generation profile. You can specify the rack heat and flow in three different ways, though ultimately the model proper uses heat generation and flow rate. Hitting OK stores these values and places the object in the room.

Flow/Heat Profile GUI

Note that each of the property GUIs had at least one button with "__ Profile". This is how the specific heat, temperature, or flow profile of an object in the room is defined. Pressing the button opens up the Flow Profile GUI:



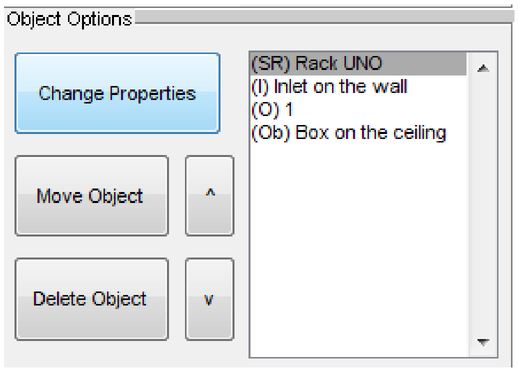
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0

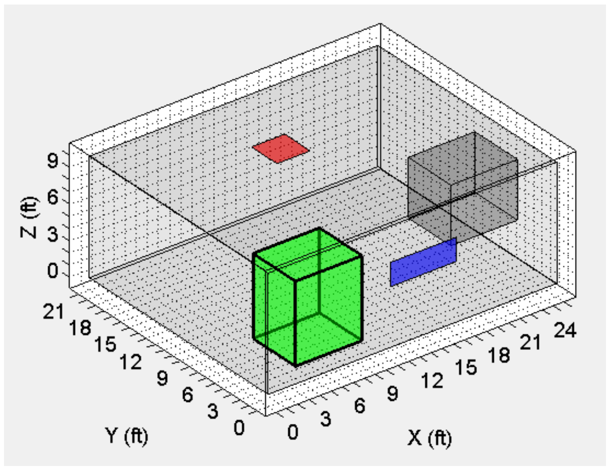
This allows you to specify the flow rate or heat generation or temperature rise profile, depending on which button was pressed to arrive here. Values can be changed manually, but there are also a variety of options to fill in the profile. Uniform, centered (1/r or 1/r^2), or custom (see explanation in Appendix) distributions can be applied, either to the profile as a whole or for individual rows and columns. You can also normalize those values to sum or average to a specified number, and there is also a “Stats” to check the sums and means of rows/columns to make sure your numbers make sense. Hitting OK stores this profile in the flow object.

- Important note: Inlet flows are positive if entering the room, outlet flows are positive if exiting the room.

Object List

The list box to the right is the object list. Each flow object currently located within the room has an entry, and when that list box entry is selected, the corresponding graphical representation of the object is highlighted:





Flow objects are grouped and prefixed by type: server racks (SR), inlets (I), outlets (O), obstacles (Ob), and partitions (P). The objects can be named in the property-editing GUI, but if they aren’t, they simply get a number corresponding to their position in the list. If the room does not contain any objects, the list has a single entry named “<none>”.

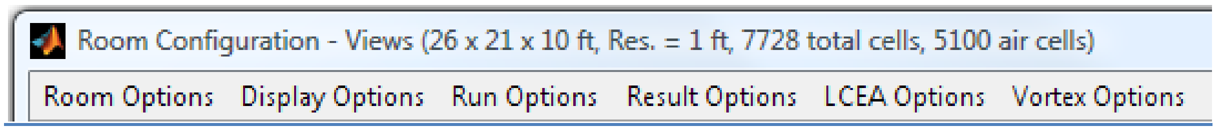
Though this does not have any effect on the actual room configuration, it is also possible for organization’s sake to change the list position of an item in the object list. The up and down arrows will push an entry up or down within its flow object group (you can’t send an inlet above a server rack, etc.)

Editing/Removing Objects

Both the physical position and the properties of an object can be edited. Selecting the object in the Object List and press the “**Change Properties**” button will call up the property editor buttons as seen earlier, only with the previously existing values filled in. Pressing the “**Move Object**” button will reopen the position-setting GUI with the rectangle set to the current position of the object. Note that if the dimensions of the object are also changed, it will reset the heat/flow profile of the object and you will have to specify it again. The "**Delete Object**" button simply removes the flow object from the room and updates the display to reflect this.

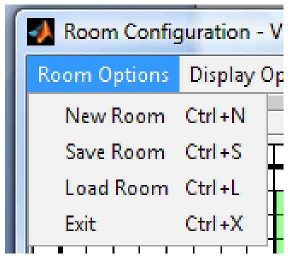
Main GUI menus

The menu bar at the top of the main GUI contains several essential and useful submenus. The main GUI title also contains some basic information about the current room, including resolution, dimensions, number of cells and number of air cells (omits those cells inside of server racks or obstacles).



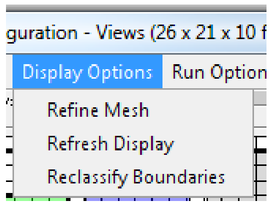
Room Configuration - Views (26 x 21 x 10 ft, Res. = 1 ft, 7728 total cells, 5100 air cells)
Room Options
Display Options
Run Options
Result Options
LCEA Options
Vortex Options

Room Options

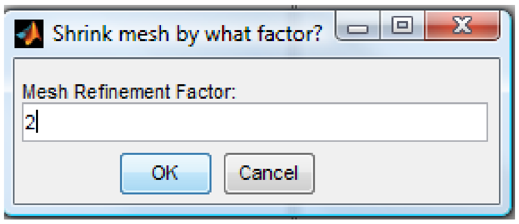


- **New Room**: Closes the current room and re-runs "makeroom.m" to create a new room.
- **Save Room**: Opens a dialog, defaulted to the "Saved Configurations" subfolder, to store a mat-file containing the current Room object.
- **Load Room**: Opens a dialog, defaulted to the "Saved Configurations" subfolder, to load a mat-file containing a Room object into the workspace, closing the current room and opening the new room in the main GUI.
- **Exit**: Closes the current GUI.

Display Options



- **Refine Mesh**: Changes the resolution of the current room, but preserves the position and flow/heat values of the currently placed objects. A dialog is opened asking for the factor by which to increase the resolution:

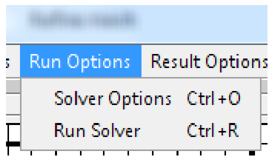

Shrink mesh by what factor?

Mesh Refinement Factor: 2

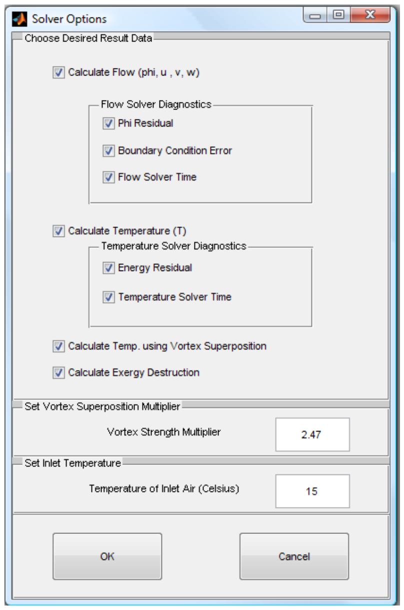
A mesh refinement factor of 2 applied to an original resolution of 0.25 meters would result in a new resolution of 0.125 meters. This can also work in reverse: a mesh refinement factor of ½ would bring the resolution back to the original value. One should be careful if coarsening the mesh, because the dimensions of the flow object may change if they do not conform to the new grid. In either case, a new room is opened with the new resolution, and each object is transferred over.

- **Refresh Display**: Simply refreshes the display, wiping and recreating the graphics objects associated with each flow object stored in the room. Useful in case a stray command accidentally overwrites one of the views.
- **Reclassify Boundaries**: This checks the flow rate through all the inlets and outlets. If the mean flow rate is negative, it reclassifies the object into its opposite class. Example: An inlet's flow profile has a mean flow rate of -1 ft/s, so it's changed to an outlet and all values are multiplied by -1. The main GUI view and object list are also updated to reflect this.

Run Options



- **Solver Options**: This opens a GUI which allows the user to set all relevant solver values.


Solver Options

Choose Desired Result Data

- [✓] Calculate Flow (phi, u , v, w)
  - Flow Solver Diagnostics
    - [✓] Phi Residual
    - [✓] Boundary Condition Error
    - [✓] Flow Solver Time
- [✓] Calculate Temperature (T)
  - Temperature Solver Diagnostics
    - [✓] Energy Residual
    - [✓] Temperature Solver Time
- [✓] Calculate Temp. using Vortex Superposition
- [✓] Calculate Exergy Destruction

Set Vortex Superposition Multiplier

Vortex Strength Multiplier: 2.47

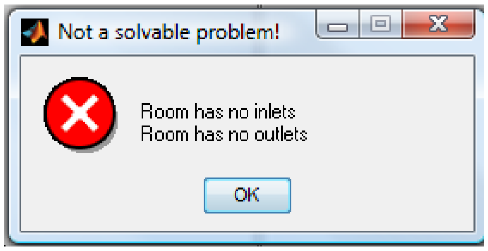
Set Inlet Temperature

Temperature of Inlet Air (Celsius): 15

The top panel allows the user to set which results they want generated by the solver, and the bottom panels set the vortex strength multiplier (see Appendix) and the inlet air temperature. The inlet air temperature is the default temperature of air flow into the room. Individual inlets can be edited to have different flow temperatures in their property editor GUIs.

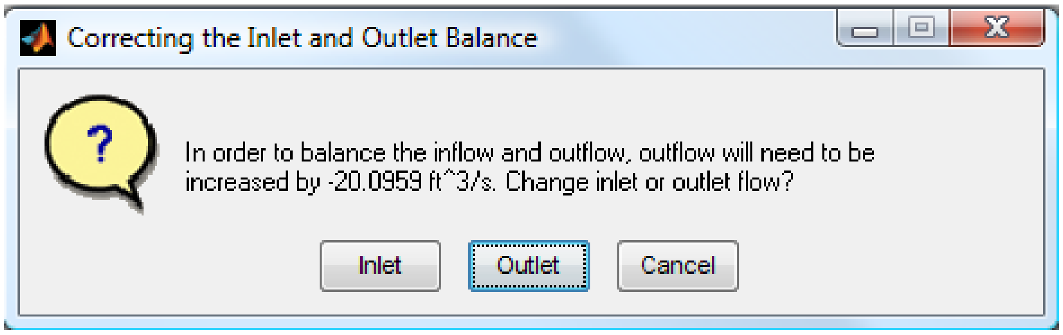
If a field/checkbox is disabled or is not relevant, it is grayed out and uneditable.

- **Run Solver**: This actually runs the flow and/or heat models, and generates a Results-class object containing the generated data. First, a check is done to ensure the room is solvable. If not, a dialog pops up with problems, such as the following:

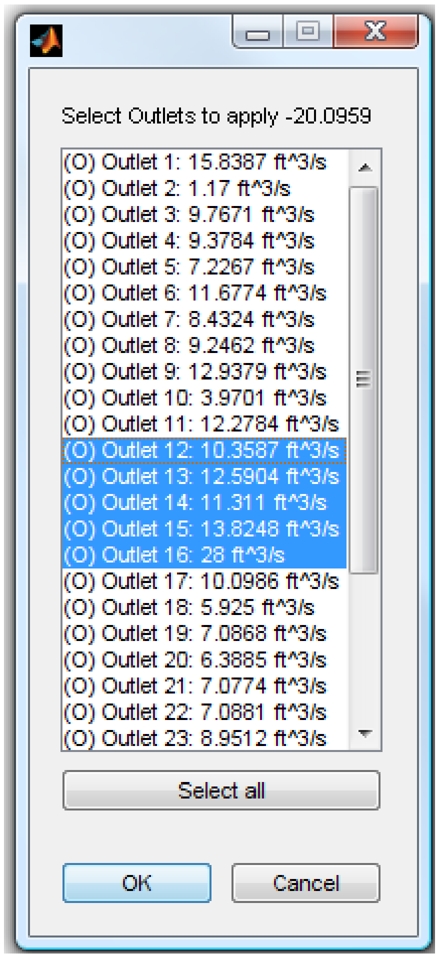

Not a solvable problem!

Room has no inlets
Room has no outlets

If there are no problems, the program then checks the flows in and out of the room to ensure they are balanced. If a larger than 0.5% change in the outflow is needed to balance them, then a dialog box is opened asking whether inlet or outlets should have their flow adjusted to rebalance. After the choice is made, a list of the current inlets or outlets in the room will appear, and the flow will be altered evenly across whichever object or group of objects is chosen. If "Cancel" is pressed instead, COMPACT will apply the flow balancing quantity to the last outlet listed in the room.


Correcting the Inlet and Outlet Balance

In order to balance the inflow and outflow, outflow will need to be increased by -20.0959 ft^3/s. Change inlet or outlet flow?


Select Outlets to apply -20.0959

(O) Outlet 1: 15.8387 ft^3/s
(O) Outlet 2: 1.17 ft^3/s
(O) Outlet 3: 9.7671 ft^3/s
(O) Outlet 4: 9.3784 ft^3/s
(O) Outlet 5: 7.2267 ft^3/s
(O) Outlet 6: 11.6774 ft^3/s
(O) Outlet 7: 8.4324 ft^3/s
(O) Outlet 8: 9.2462 ft^3/s
(O) Outlet 9: 12.9379 ft^3/s
(O) Outlet 10: 3.9701 ft^3/s
(O) Outlet 11: 12.2784 ft^3/s
(O) Outlet 12: 10.3587 ft^3/s
(O) Outlet 13: 12.5904 ft^3/s
(O) Outlet 14: 11.311 ft^3/s
(O) Outlet 15: 13.8248 ft^3/s
(O) Outlet 16: 28 ft^3/s
(O) Outlet 17: 10.0986 ft^3/s
(O) Outlet 18: 5.925 ft^3/s
(O) Outlet 19: 7.0868 ft^3/s
(O) Outlet 20: 6.3885 ft^3/s
(O) Outlet 21: 7.0774 ft^3/s
(O) Outlet 22: 7.0881 ft^3/s
(O) Outlet 23: 8.9512 ft^3/s

After this, the solver runs. The command line displays the progress of the solver:

```
-----Flow Solver START
Flow Solver: Initialization finished at 0.057766 seconds
Flow Solver: Creating nodal interrelations
Flow Solver: Nodal interrelations tabulated at 0.11872 seconds
Flow Solver: Population of matrix finished at 0.99185 seconds
Flow Solver: Matrix solved at 1.1037 seconds
Flow Solver: Final results recorded at 1.202 seconds
-----Flow Solver END
```

If the Temperature solver was not enabled in the solver options, the program stops here, generating a result data object, exporting it to the workspace as “FlowResults”, and opening the Results GUI (explained later). Otherwise, the program moves on to the temperature solver after checking for the presence of heat generation. In the temperature solver, as with the flow solver, the command line gives its status:

```
-----Temp Solver START
Temp Solver: Initialization finished at 0.077155 seconds
Temp Solver: Nodal interrelations tabulated at 0.16708 seconds
Temp Solver: Population of matrix finished at 0.46946 seconds
Temp Solver: Matrix solved at 0.53894 seconds
Temp Solver: Final results recorded at 0.54408 seconds
-----Temp Solver END
```

This is COMPACT 1.0, without the added vortex solver. If the temperature solver using vortex superposition was enabled, then it uses the initial temperature field to solve for a vortex field and again for temperature:

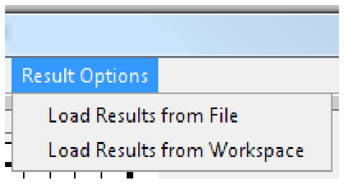
```
-----Vortex Solver START
Vortex Solver: Initialization finished at 0.032825 seconds
Vortex Solver: Input translation finished at 0.035118 seconds
Vortex Solver: Vortices characterized at 0.035975 seconds
Vortex Solver: Flow field created at 1.9204 seconds
Vortex Solver: Final results recorded at 1.9215 seconds
-----Vortex Solver END

-----Temp Solver START
Temp Solver: Initialization finished at 0.077463 seconds
Temp Solver: Nodal interrelations tabulated at 0.16673 seconds
Temp Solver: Population of matrix finished at 0.46717 seconds
Temp Solver: Matrix solved at 0.53237 seconds
Temp Solver: Final results recorded at 0.53751 seconds
-----Temp Solver END
```

Finally, if exergy destruction was chosen as a desired output of COMPACT, the exergy destruction code is run using the latest temperature field:

```
-----Exergy Solver START
Exergy Solver: Initialization finished at 0.058087 seconds
Exergy Solver: Exergy destruction field created at 0.3505 seconds
Exergy Solver: Final results recorded at 0.35152 seconds
-----Exergy Solver END
```

All these solver outputs track the amount of time used to accomplish individual subsections of the solver. Once the solver has finished, the results are stored in a Result-class object and then opened by the Results GUI (see next section).

Result Options



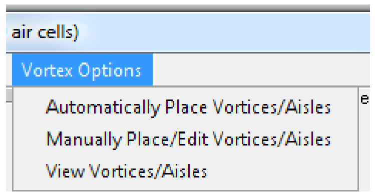
There may be cases when it is desirable to directly access Results-class objects without generating them from a room. The “Result Options” menu loads a selected Results object into the Results GUI. In both cases, the main GUI opens the room that the results were generated from.

- **Load Results from File**: Opens a dialog, defaulted to the “Generated Results” subfolder, to load a mat-file containing a Results object into the workspace and open the Results GUI.
- **Load Results from Workspace**: Opens a dialog listing all the Results-class objects currently residing in the base MATLAB workspace, to open the selected object into the Results GUI.

LCEA Options

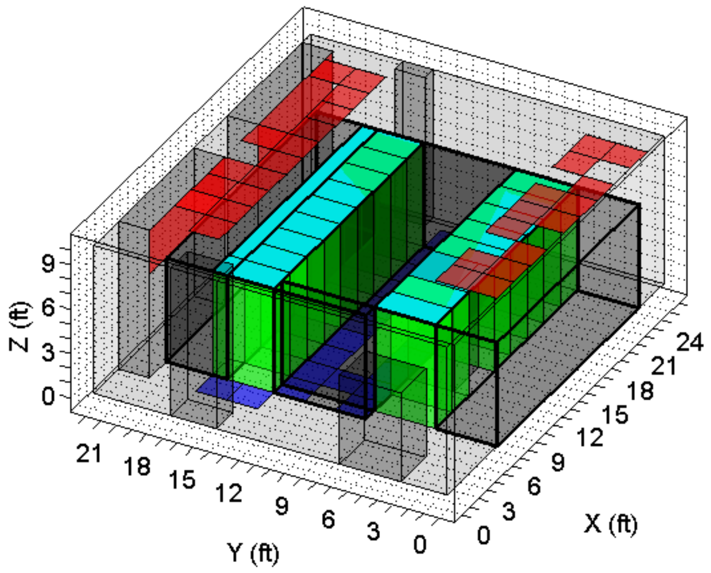
This section is in progress, does not do anything yet. May be used to link to DLCEA.

Vortex Options


air cells)
Vortex Options
Automatically Place Vortices/Aisles
Manually Place/Edit Vortices/Aisles
View Vortices/Aisles

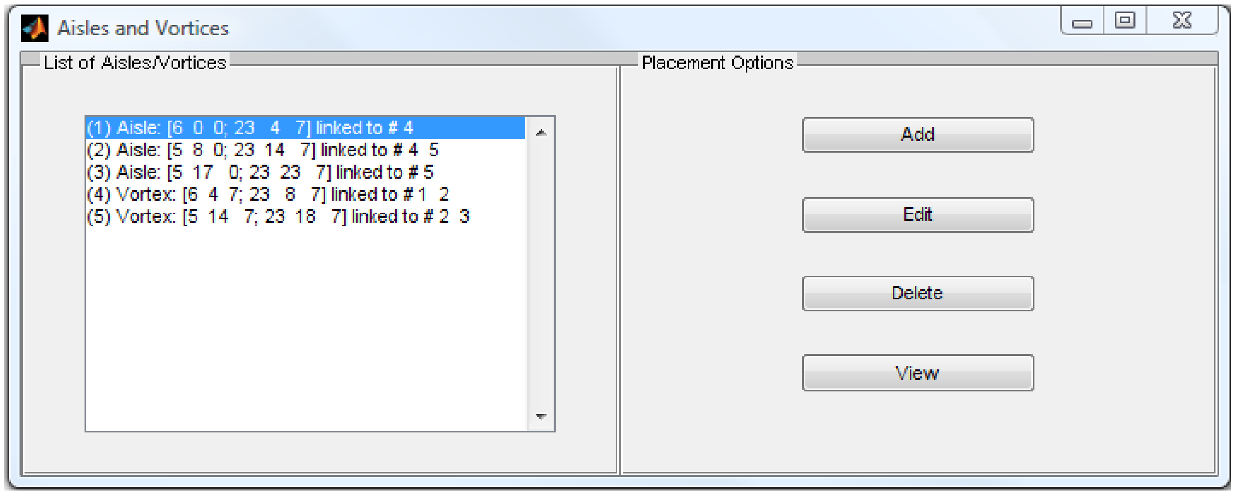
These options are associated with the vortex superposition code in COMPACT, which is used to correct temperature results generated by the basic model. To see a more in-depth explanation of the vortex superposition method, see the Appendix.

- **View Vortices/Aisles**: This simply shows the locations of vortices (located atop rows of server racks) and aisles (spaces in front and behind these rows). A figure is created showing the existing 3-D view of the room with the aisle and vortex positions overlaid.



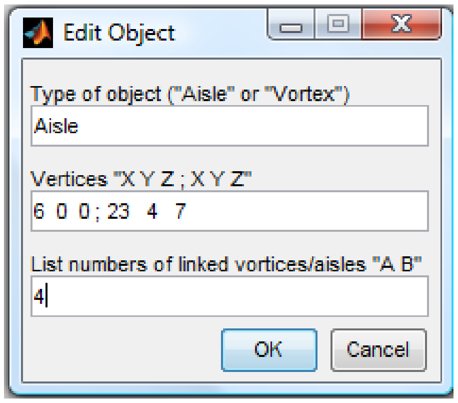
- **Automatically Place Vortices/Aisles**: This option automatically defines aisles in the room and places vortices on top of rows of server racks. If vortices and aisles have already been defined in the room, a dialog will ask you if you want to overwrite this information. ViewVortex.m will be called afterwards to show the position of the automatically placed vortices/aisles.
- **Manually Place Vortices/Aisles**: Automatic placement may not always work for highly atypical room configurations, so an option is available to manually place or edit the locations of vortices

and aisles. Hitting “View” does the same thing as “View Vortex”, but only for the individual selected aisle of vortex.


Aisles and Vortices

List of Aisles/Vortices

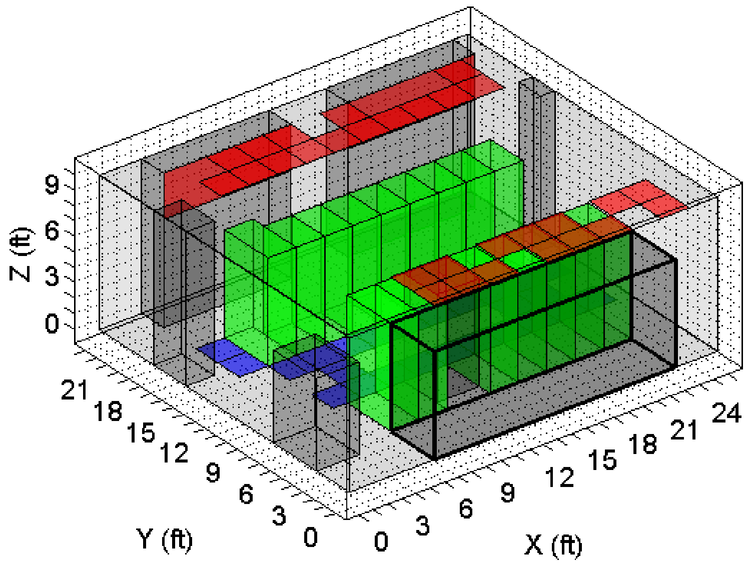
(1) Aisle: [6 0 0; 23 4 7] linked to # 4
(2) Aisle: [5 8 0; 23 14 7] linked to # 4 5
(3) Aisle: [5 17 0; 23 23 7] linked to # 5
(4) Vortex: [6 4 7; 23 8 7] linked to # 1 2
(5) Vortex: [5 14 7; 23 18 7] linked to # 2 3


Edit Object

Type of object ("Aisle" or "Vortex")
Aisle

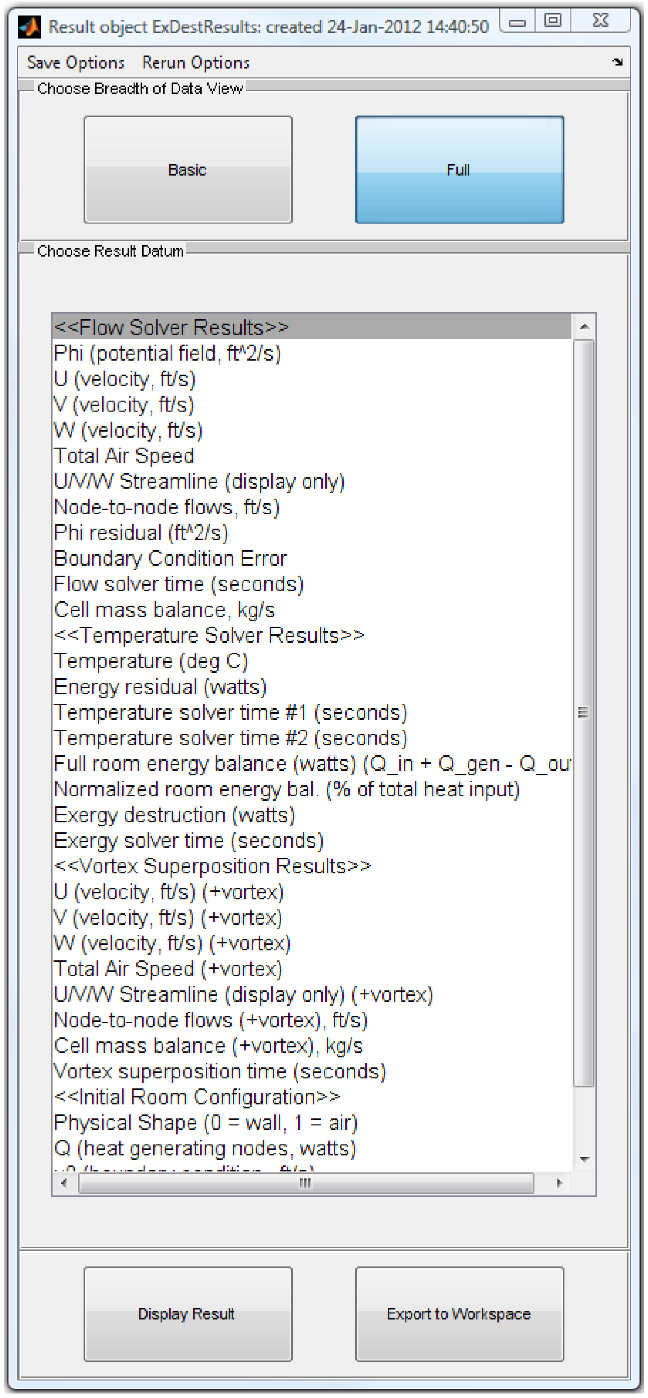
Vertices "X Y Z ; X Y Z"
6 0 0 ; 23 4 7

List numbers of linked vortices/aisles "A B"
4



Results GUI

The Results GUI processes the Results object, and lists various relevant values/arrays. There is also the option of exporting data to the workspace, and visualizing 3-D scalar fields.



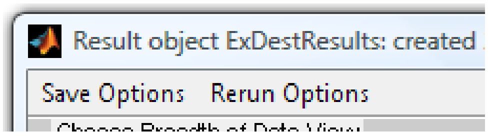
<<Flow Solver Results>>
Phi (potential field, ft^2/s)
U (velocity, ft/s)
V (velocity, ft/s)
W (velocity, ft/s)
Total Air Speed
U/V/W Streamline (display only)
Node-to-node flows, ft/s)
Phi residual (ft^2/s)
Boundary Condition Error
Flow solver time (seconds)
Cell mass balance, kg/s
<<Temperature Solver Results>>
Temperature (deg C)
Energy residual (watts)
Temperature solver time #1 (seconds)
Temperature solver time #2 (seconds)
Full room energy balance (watts) (Q_in + Q_gen - Q_ou
Normalized room energy bal. (% of total heat input)
Exergy destruction (watts)
Exergy solver time (seconds)
<<Vortex Superposition Results>>
U (velocity, ft/s) (+vortex)
V (velocity, ft/s) (+vortex)
W (velocity, ft/s) (+vortex)
Total Air Speed (+vortex)
U/V/W Streamline (display only) (+vortex)
Node-to-node flows (+vortex), ft/s)
Cell mass balance (+vortex), kg/s
Vortex superposition time (seconds)
<<Initial Room Configuration>>
Physical Shape (0 = wall, 1 = air)
Q (heat generating nodes, watts)

Also note that the time and date at which the results were generated is shown in the figure title. The buttons at the top of the list of results control the breadth of the data view. Either only basic results can be shown, or the full range of results possible to extract from the input object. Such results include, but are in no means limited to:

- The potential field and extracted velocity fields
- The temperature field
- The energy balance residuals
- The exergy destruction field
- Solver times
- Some room configuration values, like the location of boundary conditions and heat nodes
- The full room energy balance - the heat generation of the server racks is subtracted from the convective heat transport through the outlets. Ideally this value should be close to zero.

The model can easily add more types of results within the code; consult “ListResultsGUI.m” if you wish to do so.

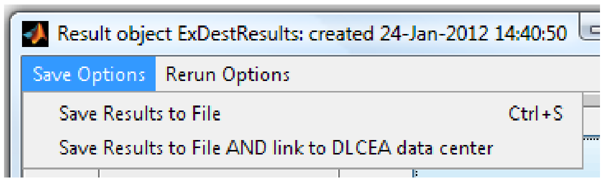
Result GUI Options



There are two menus in the results GUI: Save options for storing the results, and Rerun options for running the solver off of results (i.e. find temperature from flow solver results, find vortex-superposition corrected temperature from basic modeled temperature)

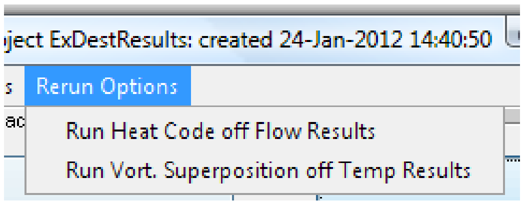
- Note: The rerun options have some bugs and are also a low priority for correction. It is advisable to just rerun the whole code from the main GUI.

Result Save Options


Result object ExDestResults: created 24-Jan-2012 14:40:50
Save Options
Rerun Options
Save Results to File Ctrl+S
Save Results to File AND link to DLCEA data center

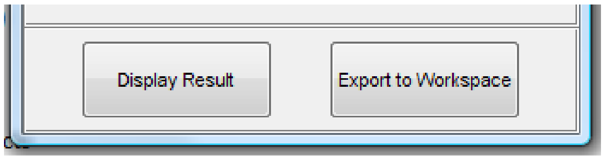
- **Save results to file**: This opens a dialog, defaulted to the “Generated Results” subfolder, to store a mat-file containing the object corresponding to the currently displayed result data. The results can be re-opened from the main GUI if desired.
- **Save results to file and link to DLCEA data center**: This performs identical actions to the first option, but then searches the base MATLAB workspace for data center objects associated with Dave Lettieri’s LCEA code. If such objects are found, they are listed in a dialog box for the user to pick. The chosen object has the exergy field and the file location of the full results object (just-saved) stored in its “exair” and “roomloc” properties.

Result Rerun Options


ject ExDestResults: created 24-Jan-2012 14:40:50
Rerun Options
Run Heat Code off Flow Results
Run Vort. Superposition off Temp Results

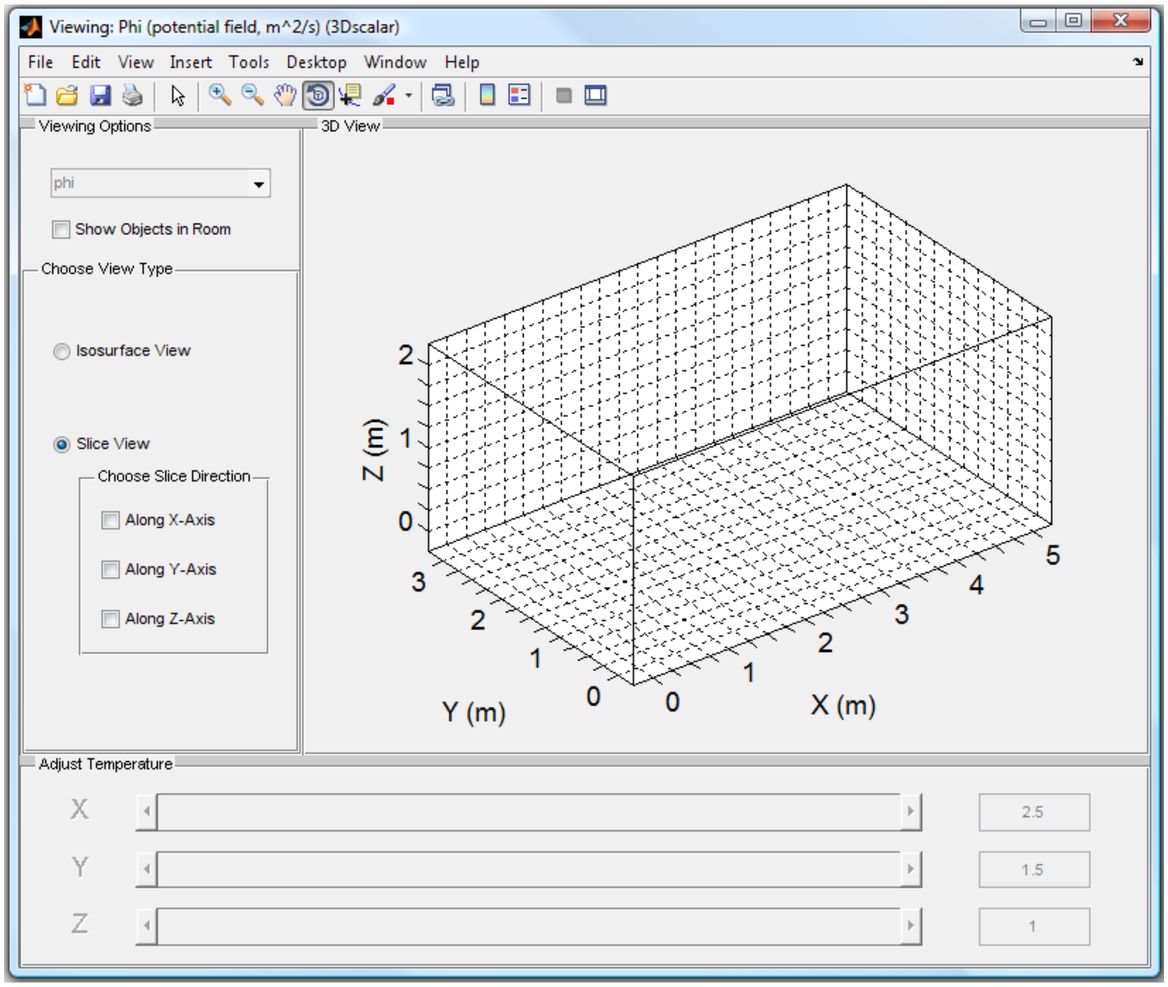
- **Run Heat Code off Flow Results**: This takes the flow-related portions of the current results, and runs the temperature solver. The solver options GUI pops up to allow changes to the temperature solver before running.
- **Run Vort. Superposition off Temp Results**: This takes the temperature results from the current results and runs the vortex superposition and temperature code, overwriting the existing temperature field.

Result GUI Buttons



- **Export to workspace**: This takes the raw data associated with a result and exports it to the base MATLAB workspace. For example, if “Temperature” is selected, pressing this button creates a variable ‘T’ in the workspace. If the full room energy balance is selected, a variable ‘Q_bal’ is assigned to the workspace.
- **Display results**: This button executes different functions depending on the associated result entry that was selected when pressed. The results are divided into five different methods of display:
    - Scalar (solver time, full room energy balance) – A simple dialog pops up, displaying the scalar value.
    - 2D evolution (no current results used, but the option is there) – a plot is generated, with the x-axis labeled “Iterations”.
    - 3D vector (streamline) – A new figure is created, with streamlines traced from the inlets and out-flowing faces of server racks.
    - 3D scalar (temperature, potential, residuals) – creates a special scalar field display GUI with isosurface and slice views
    - 4D scalar (flows, boundary condition errors) – actually a series of closely related 3D scalar fields, with the option to select each from a drop-down list in the scalar field display GUI

Scalar Field Display GUI



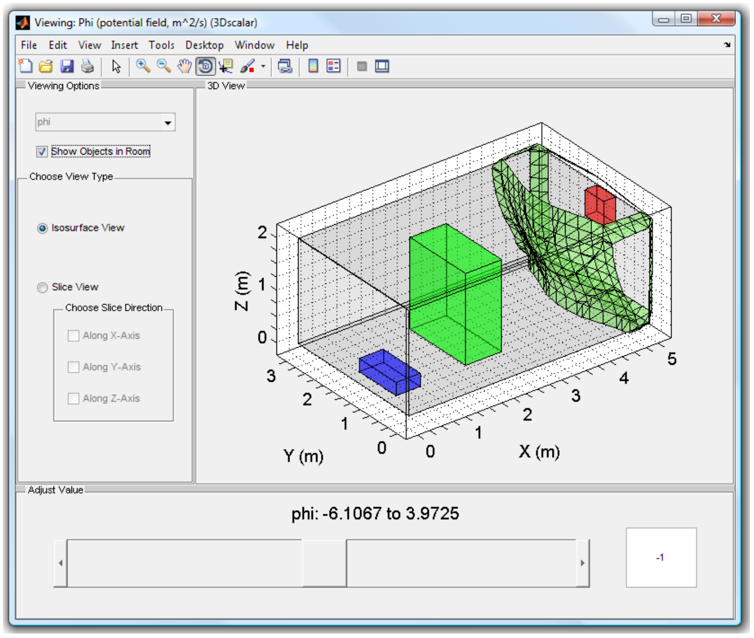
This GUI is created to display 3-D scalar fields. The figure title shows the name of the result datum and the method of display. If the input data is a single 3-D scalar field, the pop-up menu in the top left is disabled and grayed out, but if it is a 4-D scalar field, the menu contains a list of different sets of 3-D data to view, allowing the user to see one at a time. The “Show Objects in Room” checkbox on the left side of the GUI allows the user to choose whether to superimpose the flow objects onto the current view.

On the bottom are sliders, grayed out unless their associated view is selected in the “Choose View Type” panel. They are also labeled with the name of the value being adjusted. If the slider is enabled, they can be adjusted either by moving the slider, at which point the new value is displayed in the field to the right, or a new value can be entered into that field to move the slider to the appropriate location (after pressing Enter).

In the GUI, there are two methods of displaying a 3-D scalar field:

Isosurface View

The isosurface view is created by interpolating the scalar field to estimate the coordinates of surfaces of equal value. If the values are all equal, nothing will display. The slider value corresponds to the value of the isosurface, and the range of possible values is shown above it.


Viewing: Phi (potential field, m^2/s) (3Dscalar)

Viewing Options

phi

[✓] Show Objects in Room

Choose View Type

[● Isosurface View]

○ Slice View

Choose Slice Direction

☐ Along X-Axis

☐ Along Y-Axis

☐ Along Z-Axis

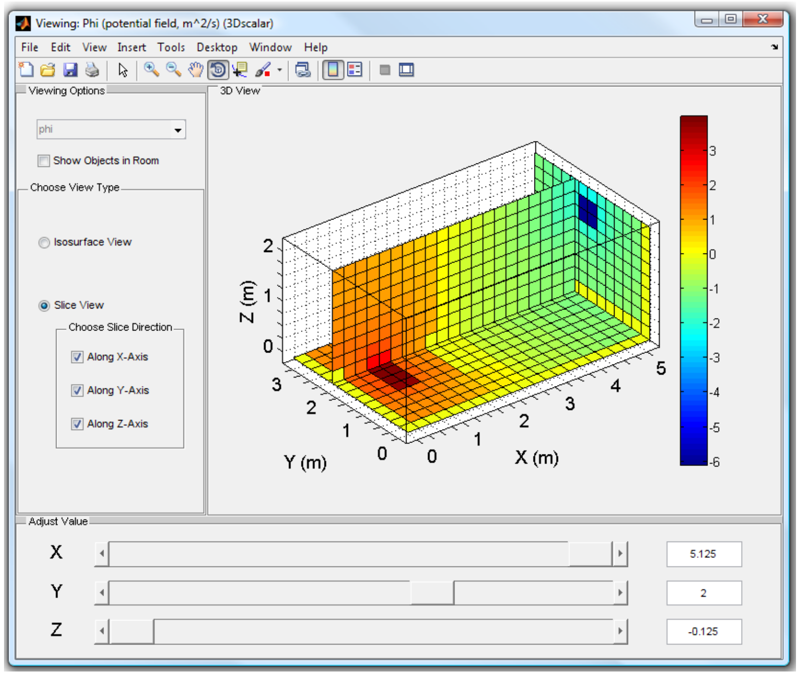
Adjust Value

phi: -6.1067 to 3.9725

-1

Slice Views

The slice view is created by interpolating the field values along a plane through the room, and displaying a color-mapped plot on that plane. Any combination of the x, y, and z plane slices can be displayed concurrently.


Viewing: Phi (potential field, m^2/s) (3Dscalar)

Viewing Options

phi

☐ Show Objects in Room

Choose View Type

○ Isosurface View

[● Slice View]

Choose Slice Direction

[✓] Along X-Axis

[✓] Along Y-Axis

[✓] Along Z-Axis

Adjust Value

X 5.125

Y 2

Z -0.125

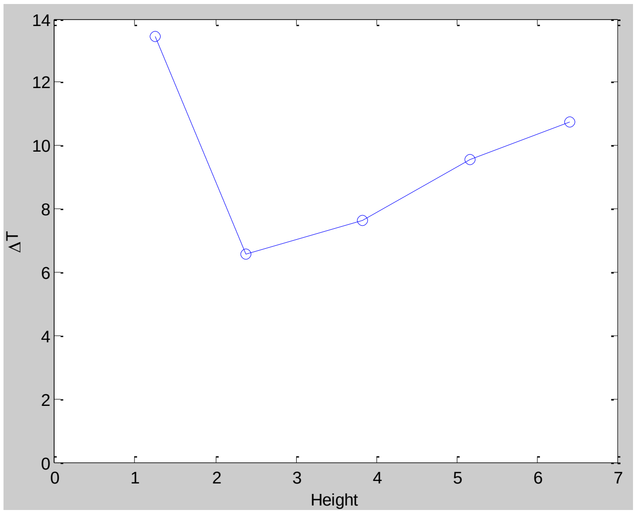
Appendix

Custom distributions

Say you have the thermistor locations and temperature rises at only 5 spots, but there are 7 cells to place values in:

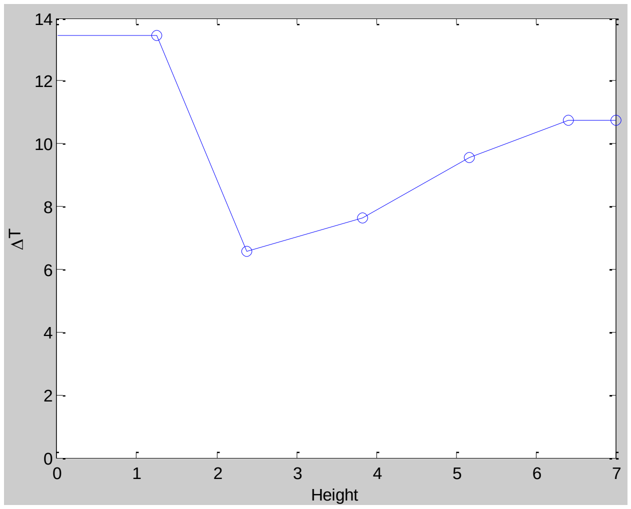
Rack_coord = [1.25 2.375 3.8333 5.1667 6.4167]

Rack_val = [13.435 6.5834 7.6529 9.5547 10.731]

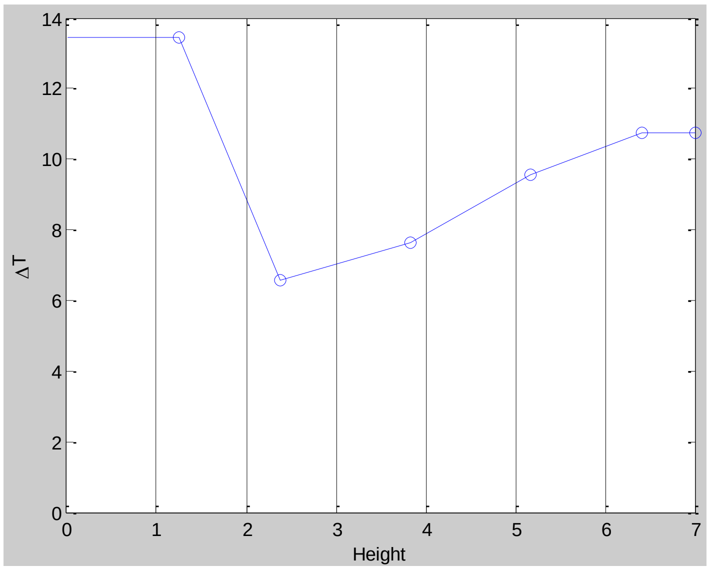


First, there's some missing information. X-axis covers the height of the rack, so what about the temperature rise at the very bottom and top of the rack?

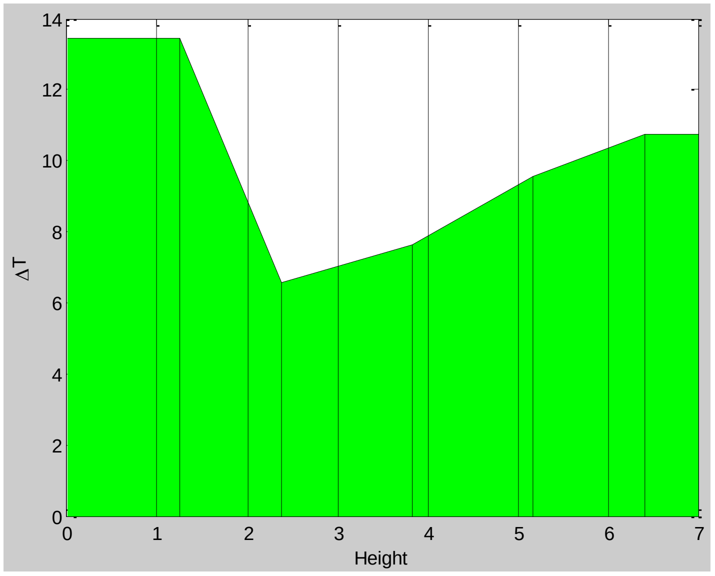
Let's extend the temperatures to the top and bottom of the rack. Since we are dealing with temperatures, this means keep the temperatures the same. If it was heat generation, we would set the edges to zero heat instead, so there would be a steep decline to 0 at x=0 and x=7.



Now let's divide the rack into its individual cells.



Visualize this as a series of trapezoids:



Average out the area under the trapezoids to find an average temperature for each cell, and use that value.

Vortex superposition

Vortex superposition was developed as a method for correcting excessively high temperatures which appeared in the original model. This was judged to be due to lack of accounting for buoyancy effects. Essentially, the temperature field from the original model is used to create a corrective flow field (which happens to contain vortex velocity fields), and the temperature code is run once more.

Theory + Implementation

The model works as follows: first, it calculates the average temperatures in the hot and cold aisles and the entire room. From this, we assume the Boussinesq approximation applies, in which the buoyancy force is an added term in the momentum equations, but incompressibility is still assumed in mass continuity. The buoyancy force is based on density variations, which is linked to differences in average temperature (Eq. 1a,b). δm refers to the mass of a heated parcel of air.

$$F_{\delta m} = \delta m(\Delta\rho/\rho_0)g \quad (1a)$$
$$\Delta\rho/\rho_0 = \beta\Delta T \quad (1b)$$

We focus on a single of row of racks at a time, consisting of a cold aisle, a hot aisle, and the row of racks itself. We then assume that in the hot aisle, the buoyancy force is applied over some distance on the order of the server rack height. For this model, we assume that height is one-half of the server rack height. Next, we assume that this force applied over the height is used entirely to accelerate the air upwards; in other words, the added kinetic energy from buoyancy forces (Eq. 2b) is equal to the added kinetic energy of vertical air flow (Eq. 3).

$$\Delta KE = F_{\delta m}H = \delta m(\beta\Delta T)gH \quad (2a)$$
$$\Delta KE = \delta m\beta\left(T_{H,avg} - T_{room,avg}\right)gH \quad (2b)$$

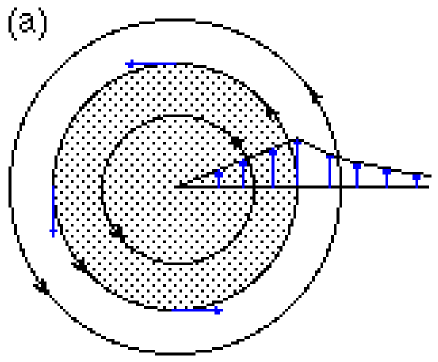
β is the volumetric thermal expansion coefficient of the air, which can be approximated as 1/T. $T_{H,avg}$ and $T_{room,avg}$ are the average hot aisle temperature and the overall average room temperature, respectively.

$$\Delta KE = \frac{1}{2}\delta m w_B^2 - \frac{1}{2}\delta m w_{H,avg}^2 \quad (3)$$

By equating the two terms, we find the added vertical air velocity due to buoyant forces:

$$w_B = \sqrt{w_{H,avg}^2 + 2g\beta\left(T_{H,avg} - T_{room,avg}\right)H} \quad (4)$$
$$\Delta w_B = w_B - w_{H,avg} \quad (5)$$

w_B and $w_{H,avg}$ are the average upward hot aisle velocity with and without buoyant forces. The latter is computed from the COMPACT 1.0 model. This process is repeated for each row of server racks.



Now, for this row of server racks, we superimpose a Rankine vortex on the top of the row. A Rankine vortex is acts as a solid-body (rotational) vortex within its core, and a potential (irrotational) vortex outside of that core. The velocity profile of a Rankine vortex can be seen in Eq 6 below:

$$v_\theta = \begin{cases} k_V r, & r < R \\ k_V R^2/r, & r \geq R \end{cases} \quad (6)$$

A benefit of this method is that it spreads out the area of high recirculation, while adding the tendency for warm air to rise, yet preserving the potential flow model outside of the vortex core, as potential vortices satisfy Laplace's equation. Also, the effect of the vortices drops with distance from the tops of the server racks, limiting the corrective flows to the areas of greatest concern.

The distance from the vortex center is given by r, the center and R (the core radius) is set to half the depth (the distance from front to back) of the row of server racks. The center of the vortex is also centered on the top of the row of server racks, so the core covers the entire top surface. The quantity k_V is chosen so that v_θ at the edge of the vortex core (aka the cell right above the warm air outlet of the server rack) is proportional to the added velocity Δw_B previously calculated.

$$k_V = V_s * \Delta w_B / R \qquad (7)$$

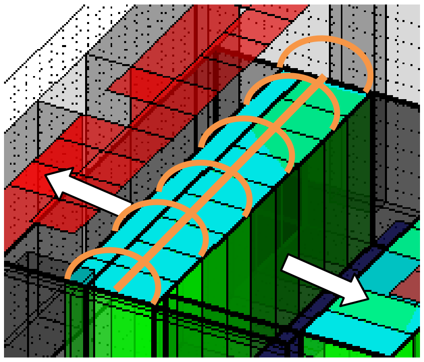
V_s is a vortex strength multiplier which was tuned to optimize the results of the model to most closely predict the inlet face temperatures of rows of server racks. Currently the default value is 2.47, but this can be changed in the COMPACT Solver Options GUI.

Automated Vortex Placement

Note that the hot/cold aisles and vortex location in the previous section need to be placed. Doing this job entirely manually would be onerous, but thankfully an automated system is available. Automated vortex placement works as follows:

- First, all racks and obstacles in the room are sorted into clusters of racks. Each object is checked to find if it is adjacent to (shares a face with) other racks/obstacles, and then those relations are analyzed to find groups of linked servers/obstacles.
- These groups are treated as rows of racks. The top plane of each cluster is set as the vortex location. The axis of the vortex will be perpendicular to the direction of flow through the racks, and bisect the specified upper plane. This ensures the core of the vortex completely covers the top of the row of racks.
- From each row of racks, the direction in which most of them direct flow (x or y direction) is used to define the two flow faces of the row.
- The "aisles" are extruded from these flow faces in each flow direction either until they meet a solid surface taking up >70% of the next extruded space, or until it has extended out more than 2 times the depth (in the flow direction) of the row of server racks.

A small graphical representation of COMPACT's automated vortex placement:



- Cyan plane: vortex location as represented by COMPACT; plane bisects the vortex core.
- Orange line: vortex axis, bisects vortex plane.
- Orange arcs: border of vortex core, which is where the vortex changes from solid-body rotation to irrotational.
- Arrows: direction of extrusion of aisles.

List of COMPACT functions

Below is a comprehensive list of functions used by COMPACT, as well as some diagnostic functions.

Model functions

These functions are needed to run COMPACT and the GUI (there are about 40 of these). If you can't find the function here, it's not necessary and is probably listed under the diagnostic functions.

- **Classes and class-specific functions:**
  - **@Air**
    - **Air.m:** contains the basic properties/units of air. Anywhere in the code, you can ask for "Air.cp" or "Air.units" and it'll give you the values (1003 J/kg-K and 'feet'). Allows you to switch between feet and meters by commenting out blocks of code. Needs more testing.
  - **@Boundary**
    - **Boundary.m:** Class definition file for Boundary objects, which are objects defining an inlet or outlet, or more generally any "window" in or out of the room as a whole (server racks don't count, we're not doing this FLUENT-style).
    - **boundary_properties.m:** This creates the GUI that lets you define the name and flow properties of an inlet or outlet. You can also switch between inlet/outlet types with it.
  - **@FlowObject**
    - **FlowObject.m:** This is an important one. This is an abstract superclass definition file from which all physical objects (and their classes) inside the room are derived. Boundary, ServerRack, Partition, and Obstacle are all subclasses.

    - **FlowObjProfileGUI.m:** This makes a GUI which lets you specify the values in a flow/heat/temperature profile on a face of a flow object in the room.
    - **PatchData.m:** This is a low-level function used by many other functions in drawing the room object graphics. Takes the room object vertices and spits out a couple of matrices which go into MATLAB's "patch" function.
  - **@Obstacle**
    - **Obstacle.m:** Class definition for Obstacle objects. Just big 3D objects in the room, no flow or anything. They show up gray in the display. Almost exactly like a basic FlowObject .
  - **@Partition**
    - **Partition.m:** Class definition for Partition objects. Partitions are zero-thickness flow barriers in the room. The code pretty much treats them as walls, but the virtual node is stored in hyperspace rather than inside the wall like for the room's walls.
  - **@Profile**
    - **Profile.m:** The class definition for Profiles, which are flow/heat/temp rise matrices corresponding to flow surfaces on racks or inlets/outlets. The objects store the matrices, and the type of profile (heat gen, flow, temp rise) and added info about the distribution if necessary.
  - **@Results**
    - **Results.m:** Class definition file for Results. Each property stores information generated by the model, including temperature, potential and exergy destruction fields, as well as model times, residuals, the original room input, etc. This class is also used to store the settings specifying what results the model should generate.
    - **ExergyDestruction.m:** Exergy destruction code. Takes temperature results as an input, iterates through each air cell in the room and calculates entropy of air coming in and out, uses that to compute the exergy destruction for the cell.
    - **ListResultsGUI.m:** Creates the GUI that lists all the possible results you can visualize or export to workspace. Can also save to file. Takes a Results object as an input, reads it for the available info to extract, and makes the GUI. Coded in such a way to allow easy addition of new metrics as they are thought of. Just use the addresults function, specify the title of the result that you want listed in the GUI, the data, the name if you export the data, the form of visualization to use with it, and whether it's a basic metric or a more advanced one.
    - **SolveHeatMatrix.m:** The CT in COMPACT. Heat code, obviously. Takes flow results with the potential field and computes the temperature. May be a bit hard to read, since it's heavily optimized.
    - **VortexSuperposition.m:** The vortex superposition code. Imports vortex and aisle positioning info stored in the room file, grabs the temperature field from inputted temperature results, calculates the vortex strength and outputs the flow field to superimpose, also storing it in the TempResults object under the property "VortexSuper". Currently about 240 lines, I could probably shorten it if I thought about it for a while.

- **@Room**
  - **Room.m:** The class definition file for the room as a whole. Flow objects are stored in Room objects, as well as results settings, graphical settings, and other solver options. Methods are included for adding and deleting flow objects from the room, counting the number and types of flow objects, and correcting the inflow vs. outflow values, among others.
  - **room_shape.m:** This is one of, if not the most important functions in the code. It creates the main GUI in which the room configuration is defined, and provides callbacks to essentially all the model's functions. Create/store/load rooms, place/delete/move/edit objects, set solver options and run the solver, refine the mesh, refresh the display, load results from file. Refers to most of the GUI functions located elsewhere in COMPACT. As a result, it's a monster, about 750 lines of code. 200 lines to make the GUI, 550 for the callbacks for all the buttons/menus. Ctrl-F is your friend.
  - **SolvePotMatrix.m:** The potential flow code, the P in COMPACT. Takes the original room object and generates a velocity potential field. Optimized like SolveHeatMatrix, but a little easier to read.
  - **AutomatedVortexPlacement.m:** The most recent awesome code to be added to COMPACT. It automates placement of aisles and vortices in a room. Sorts racks in the room into rows/clusters, and extrudes aisles out from the flow faces of these rows of racks. Outputs the vortex and aisle info as raw data for other visualization and manual editing functions to use. Actually, RowFinder.m does the sorting.
  - **CheckSolvable.m:** This function executes after ordering the solver to begin. It checks the room for solvability and returns as false if unsolvable, along with a dialog box listing all the issues preventing it. This includes negative or nonfinite resolution/dimensions, zero flow everywhere in the room, and obstruction of flow faces by other objects.
  - **DisplayResults.m:** This function is referred to by the ListResultsGUI function when the Display Results button is pressed. It checks the nature of the data being displayed and if simple, just outputs a dialog box or 2-d plot. If it's a 3-d or 4-d field, it calls on IsoSliceViewGUI or UVWstreamline functions.
  - **extract_BC_data.m:** This function is actually pretty often used. It extracts some basic boundary conditions from the Room Object, like a 3-d field of which cells are in air (room_config), and velocity boundary conditions u0,v0,w0, as well as the heat generation field Q. Also makes a partition_config, which needs to be separate since it defines a zero-depth surface and not actual cells in the room.
  - **MakeRackLinkTable.m:** This is used in the potential flow and heat flow functions. It makes a table that links the flow between the front and back of the racks, so rather than saying that the adjacent cell to a cell on the front of rack is just inside the rack, it links it to the cell on the other side.

        - **PositionSetGUI.m:** Makes the object placement GUI to place inlets/outlet/racks/etc. Used for both placing new objects and moving the position of an existing one. Uses the image processing toolbox a bit for a draggable, resizable rectangle. Returns the vertices of the object placed as well as its flow orientation if it's an inlet or outlet.
        - **Room3D.m:** This function creates the 3-D objects you see in the main GUI. It takes the flow objects stored in the Room object and creates graphics objects representing them in the "Three Dimensional View" panel in the main GUI. Also stores their handles in the Room Object, for use in other functions.
        - **RoomProjected.m:** Does the same as Room3D, only for the projected views on the left side of the main GUI.
        - **SolverOptionsGUI.m:** Creates a GUI showing the different solver options, like the results and solvers you want to run, the vortex strength multiplier, or the default inlet air temperature (you can also customize individual inlets in their property GUIs).
        - **UpdateListBox.m:** This just updates the little listbox near the bottom of the main GUI which shows all the flow objects in the room. It gets called when an object is added or deleted from the room.
        - **ValidPosition.m:** Used before adding an object to a room, this function checks the input coordinates it is given for a flow object, and sees if it is a valid location to place. If it doesn't occupy the same space as an existing flow object, and that its flow faces aren't blocked by something already in the room.
            - Note that this seems similar to the CheckSolvable function, and it is, but this one is called before placing an object and specifying its properties. For this reason, it doesn't check to see if a server rack face is blocked, because you haven't yet specified its flow direction!
        - **VortexPlacementGUI.m:** This GUI is a recent addition, and is fairly bare bones. It creates a small GUI with a list of vortices and aisles, plus their locations and which other vortices or aisles they are linked to. AutomatedVortexPlacement.m generates these values, but this GUI allows you to view their locations and manually edit their attributes. You can also add or delete vortices/aisles entirely.
    - **@ServerRack**
        - **ServerRack.m:** The class definition file for server racks. Is a subclass of FlowObject, with added properties to store the heat generation and temperature rise profiles (flow profiles and direction are already in the superclass). Also includes a handful of common quantities as dependent properties. Also stores and updates the nature of profile characterization (use flow + temp to find heat gen? Use HG + temp to find flow?)
        - **rack_properties.m:** Creates a GUI to create or edit the flow and heat characteristics of a server rack object. Like with the boundary_properties function, you can also name the object.
- **Non-class-specific functions:**

- **makeroom.m:** The beginning function that runs the model. Creates a room object, so I personally use Rm = makeroom() to start the model. Also called when you select "New Room in the main GUI.
- **Center_Fig.m:** This one is a nifty little function I use in creating all the GUIs. Figure positioning can be weird, so this function just takes the desired dimensions, checks to see that it isn't bigger than the screen and shrinks it if so, and centers the resulting figure. Output is a 4 element vector that can be directly set as the figure's Position property.
- **extract_vel_flows.m:** This function takes the velocity potential field and room configuration/resolution and converts it to a 4-D array: the value at (i,j,k,m) is the flow heading away from the (i,j,k) cell in the m direction. Values are in length/sec.
- **IsoSliceViewGUI.m:** If the result you want to view is a 3-D or 4-D scalar field, DisplayResults.m calls on this function. It makes a GUI that allows you to view the data using X, Y, and/or Z-plane slices, or as isosurfaces. Sliders at the bottom make this process easy. You can also include the room's flow objects on the plot, if you wish.
- **ViewVortex.m:** View the location of the aisles and vortices in the room. It can take the info for multiple entries as easily as just one, so it's used to both show all vortices/aisles and to just show one. Called on both in the main GUI and in the manual vortex placement GUI.
- **BorderChecks.m:** This is only used in the initialization of the potential flow solver. It creates a handful of arrays of similar size to the room configuration matrix (telling which cells are in the air), which describe whether their corresponding room cells are on a border, what amount of flow is going through that border, whether a virtual node is at a corner and thus needs special treatment since it occupies the same space as another virtual node, etc.
- **RowFinder.m:** This is a subfunction of AutomatedVortexPlacement. It takes a room and sorts the server racks into clusters of adjacent objects aka rows. This is used by its parent function to make the aisles and vortices.

Diagnostic functions and spreadsheets

These are not necessary for running COMPACT , but useful for studying. Some of these functions involve converting ANSYS, measurements, and COMPACT data formats back and forth.

- **UpdateRack.m:** You know, I dunno what that's doing there. I think it was used by some older diagnostic function. In any case, it updates the third profile out of (heat gen, temp rise, flow rate) by computing it from the other two. Already done in the ServerRack method UpdateHeatGen
- **VortexMultiplierStudy.m:** Use this to study/tune the vortex multiplier. Needs a few runs and the measurements from those runs, and a range of V_s values to try.
- **IN PROGRESS:** EVERYTHING PAST THIS LINE IS NOT YET DONE
- **SpotCheck.m:** Just something I made to spot check all the relevant flow, temp, and energy values for a single cell in the room. Made it so I could check all those values by hand, make sure there wasn't some hidden math error.
- **RoomInterpolate.m:** This function is used to
- SpotCheckingExergyDestruction.xlsx

- plotropy.m
- VortexCenteringStudy.m
- VortexCenteringStudyTemp.m
- quiverslice.m
- TimeCheck.m
- cellsincore.m
- CCLstring_generate.m
- CFD_compare_velocities.m
- fluent_profile.m
- fluent_profile_assembler.m
- generalizedUVWstreamline.m
- generate_inputs.m
- journalstring_generate.m
- measurementmarkers.m
- process_point_data.m
- run_profiles.m
- triplecompare.m
- anemomconverter.m
- extract_thermocouple_data.m
- extract_thermocouple_data2.m
- InletFacePlot.m
- ModeledPointColumns.m
- ModelMeasComparison.m
- RI_finder.m
- SHI_finder.m
- TCLocations.m
- TCLocationsPlus.m
- thermistorheights.m
- ThermistorLocations.m

- In-depth information on the biggest critical functions?
  - makeroom.m
  - room_shape.m
  - SolvePotMatrix.m
  - SolveHeatMatrix.m
  - Room.m
  - ListResultsGUI.m