Tutorial 8: Creating Superblocks

It is important to know how to merge blocks into superblock format for CFD codes that the user is developing in a research or design department. **GridPro** allows the user to convert the grid into superblock data by using the “Save: Grid as” command that randomly merges blocks of a similar property. If the user wants to control the merge process, a merge block utility is available that controls the merging routine based on the block properties, face properties, patch definitions and surfaces created in the topology builder.

What You Will Learn

- Random merge using the “Save: Grid as” function
- About the merge block utility to merge blocks based on selected properties
- Monitoring the merge results
- Setting the maximum and minimum number of blocks for the merge block routine
- Controlling the merge block process using block face properties
- Controlling the merge block process using internal surfaces
- Using previous merge block data files to recreate a similar merge with a new grid (scheduled merge)

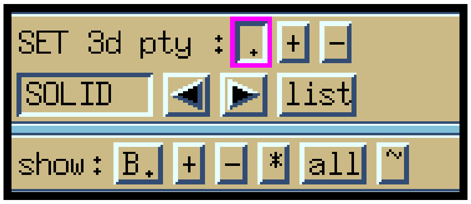
- Merge block file descriptions

Part I: Merging Blocks using “Save: Grid as”

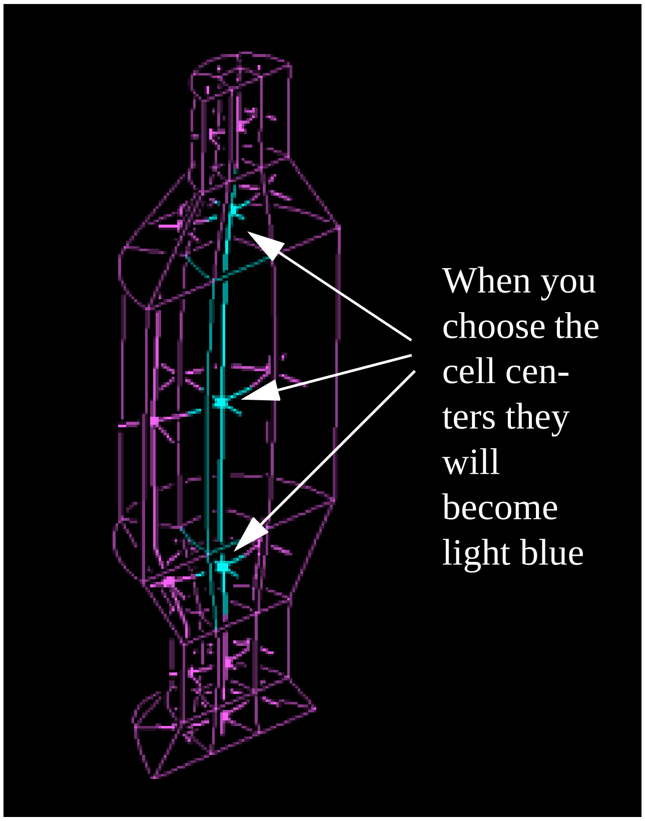
Step 1

Set Properties

A quick way to create superblocks in **GridPro** is after you have created the properties you can save the grid in merged block format. **GridPro** will automatically merge all of the blocks of the same property. Let’s look at a grid that was created in **Tutorial 7**. Load the **Tutorial_8_blk** file in the Grid Viewer and Choose PDC as your solver. Go to the **Set 3d Property** sub-command menu and choose **user2** as the property in the pull-down list.

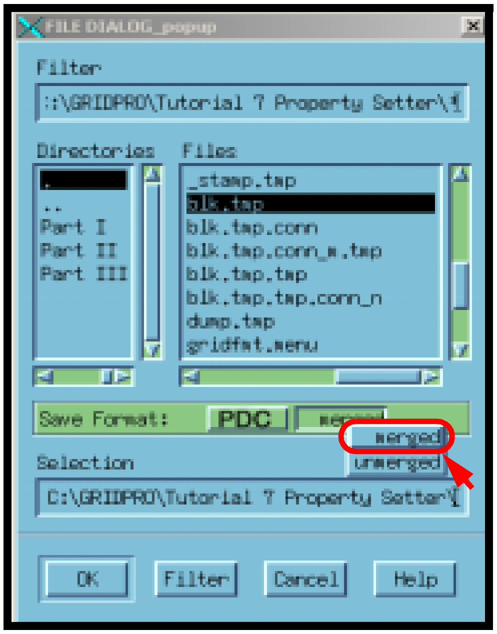


Choose the blocks using the . button and clicking on the cell centers.



When you choose the cell centers they will become light blue

Now go to the **grid** sub-menu at the top and select **save: grid as** and choose the **merged** option.

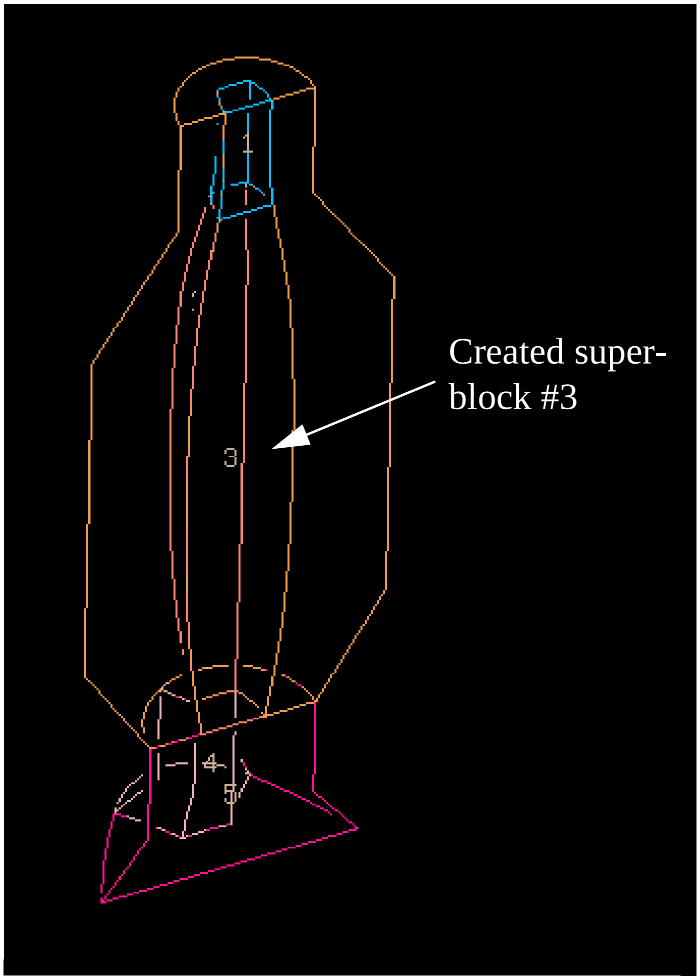


Filter: :\GRIDPRO\Tutorial 7 Property Setter\

Save Format: PDC merged

Selection: C:\GRIDPRO\Tutorial 7 Property Setter\

Reload the grid and look at the blocks, as you can see the blocks that have been set as **user2** are now merged into one superblock. All of the remaining blocks that were at the default fluid setting were also merged.



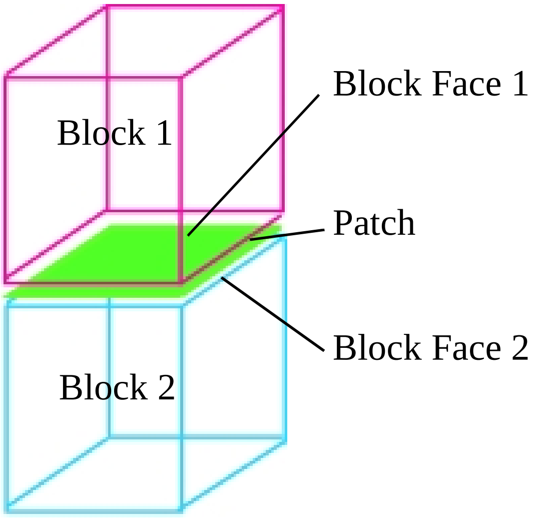
Created super-block #3

Note: the merge block routine will not merge all blocks into one superblock.

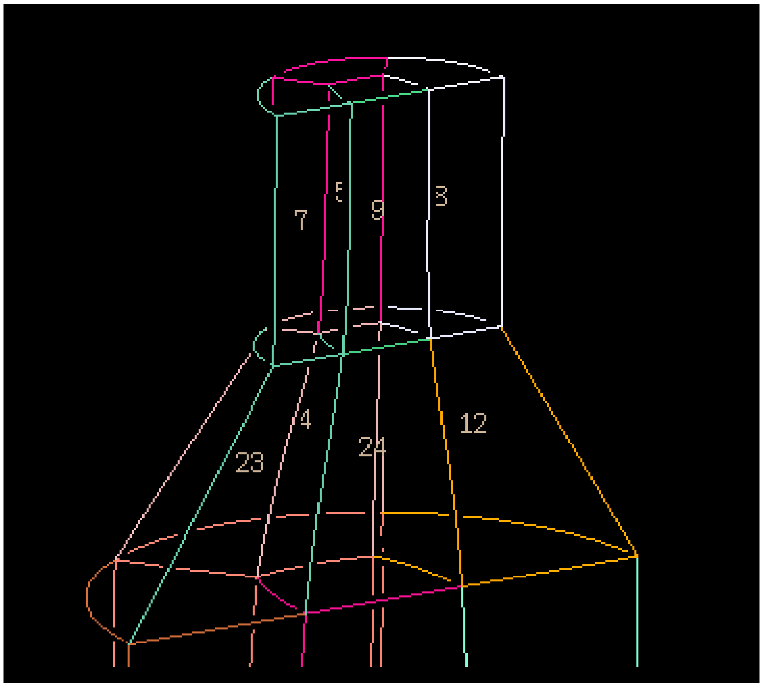
Part II: Using the Merge Block Utility

Step 1 Understanding Basic Block Structure

When a grid is created, **GridPro** creates the grid in elementary blocks based on the topology design. The user can merge these elementary blocks into larger “superblocks” based on predefined criteria and the **GridPro** algorithm developed by Dr. Rigby, Dr. Steinthorsson and W. Coirier at the NASA Lewis Research Center. Each block in **GridPro** has 6 unique **faces** and each block is connected to adjacent blocks by **patches** which function as a glue binding faces together. Two adjacent faces that belong to separate blocks will share a patch that will allow the two blocks to communicate. This connectivity functions as a “glue” and is saved into the `blk.tmp.conn` file that is automatically created as a result of the gridding algorithm. In order to merge two block faces that share a patch, the patch must be equal to or smaller than the cross sectional area of the adjacent block faces. The arrangement of the block faces and patches is shown in the picture below.



Below is a picture of elementary blocks belonging to a grid.



The connectivity information will be listed in the **blk.tmp.conn** file and will contain information on the 6 faces of each block.

To run a merge, the user must create 3 input files, and as a result of the merge, 3 output files are created. The input, resulting output files and their definitions are listed in the table below.

Table 1: Merge File Definitions

Input File	**Definition**	**Output File**	**Definition**
`file_name_blk`	Elementary block data	`file_name_blk.tmp`	Superblock data created as a result of the merge
`file_name.conn`	Elementary block data connectivity	`file_name.conn_n`	Node connectivity file for the merged blocks
`file_name.pty*`	Property information set inside the Property Setter	`file_name.conn_m`	The merge schedule file in **GridPro** format used for future merges.

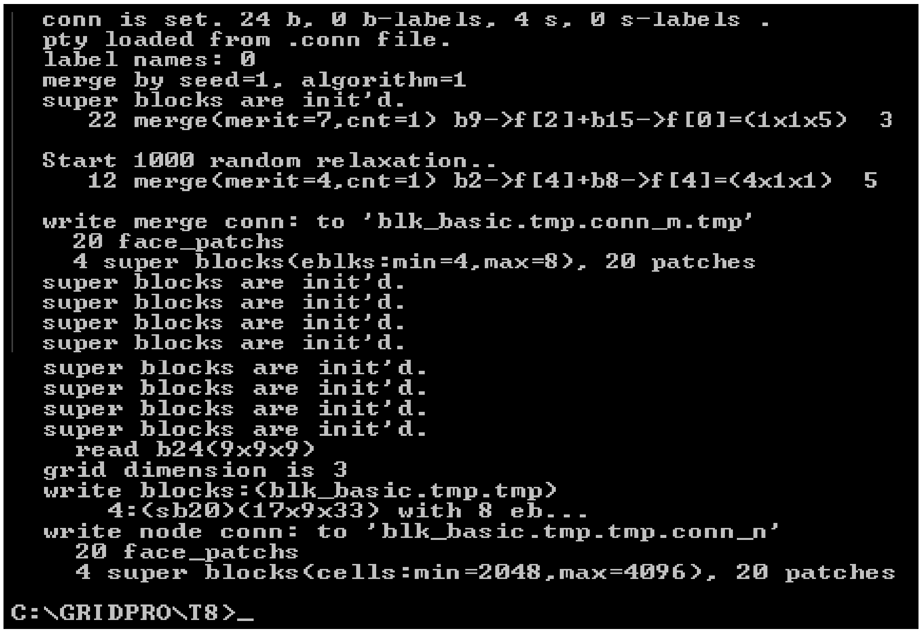
*This file is optional. If the user does not use a property file, the default value will be used. The default property for the 3D grid cells is *fluid* and *wall* for the 2D surfaces of the grid.

Step 2 Basic Merge

The basic merge command to by typed at the prompt is:

```
mrgb file_name [options] <return>
```

If we want to carry out a merge procedure in **GridPro** format based on our property file, typing in the command without the options will merge all blocks with a similar property. Let’s merge the same grid as we did in **Part I** and take a look at the results. Save the grid as **blk_basic.tmp,** start **GridPro**, load in the grid file named **Tutorial_8_blk** and enter the PDC property setter. Set the same 3 inner blocks as a **user2** property and save the grid. Two files will also be saved - the **blk_basic.tmp.pty** which contains the property information and the **blk_basic.tmp.conn** which contains the block connectivity information. Now type in the command at the prompt and hit return. The following information about the merge will output to the prompt.



```
  conn is set. 24 b, 0 b-labels, 4 s, 0 s-labels .
  pty loaded from .conn file.
  label names: 0
  merge by seed=1, algorithm=1
  super blocks are init'd.
     22 merge(merit=7,cnt=1) b9->f[2]+b15->f[0]=(1x1x5)  3

  Start 1000 random relaxation..
     12 merge(merit=4,cnt=1) b2->f[4]+b8->f[4]=(4x1x1)  5

  write merge conn: to 'blk_basic.tmp.conn_m.tmp'
    20 face_patchs
    4 super blocks(eblks:min=4,max=8), 20 patches
  super blocks are init'd.
  super blocks are init'd.
  super blocks are init'd.
  super blocks are init'd.
  super blocks are init'd.
  super blocks are init'd.
  super blocks are init'd.
  super blocks are init'd.
    read b24(9x9x9)
  grid dimension is 3
  write blocks:(blk_basic.tmp.tmp)
      4:(sb20)(17x9x33) with 8 eb...
  write node conn: to 'blk_basic.tmp.tmp.conn_n'
    20 face_patchs
    4 super blocks(cells:min=2048,max=4096), 20 patches

C:\GRIDPRO\T8>_
```

Take a look at the merge by loading the grid back into **GridPro** and you will see that it has the same superblock structure as the grid in **Part I**.

Step 3 Merge Criteria

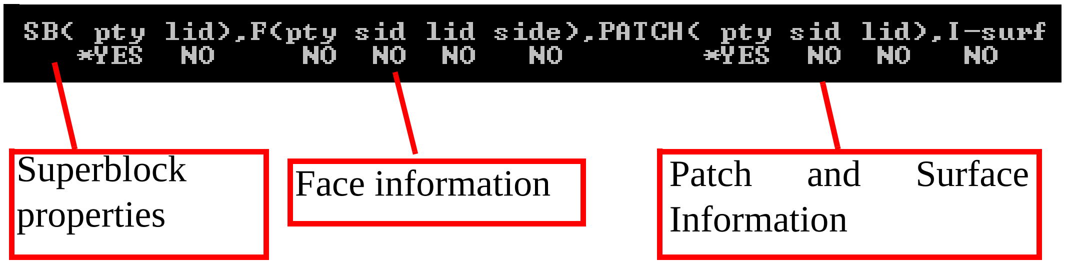
The merge routine can merge elementary blocks into super blocks based on user defined criteria. The syntax for this criteria is:

```
mrgb file_name [options] <return>
```

The four global criteria that control the merge operation are:

1) Superblock Properties
2) Faces
3) Patches
4) Surfaces created in the topology builder

When the merge routine begins, GridPro will output the merge criteria based on the user options. Below is an example of the basic merge that we conducted above.



```
SB( pty lid),F(pty sid lid side),PATCH( pty sid lid),I-surf
   *YES  NO     NO  NO  NO   NO        *YES  NO  NO   NO
```

How to control the merge routine will be explained in preceding steps, for now, a brief explanation is provided in the table below. The label id’s (lid) are not important for merging block at this time so they will be skipped.

Table 2:

Parameter	Criteria	Flag	Definition
SB (Superblocks)	pty (property)	*YES (* = Always on)	Will always preserve properties when superblocks are created
F (Face)	pty	YES	Elementary blocks need to have the same property to merge
		NO	Elementary blocks do not need to have the same property to be merged
	sid (surface id)	YES	Will merge blocks only if they share the same surface
		NO	No restrictions on whether the blocks share the same surface
	side	YES	Cannot merge faces without a unique side (unique - entirely one sided or both sides have blocks cannot have a mix on one face, gasp cannot mix)
		NO	Can merge a face without a unique side
PATCH	pty	*YES	Only merges patches with the same property
	sid	YES	Will merge blocks only if the patches share the same surface
		NO	No restrictions on whether the blocks share the same surface.
I-surf (Internal Surface)		YES	Will not merge through internal surfaces
		NO	Will merge through internal surfaces

Note: that surfaces are defined as those associated with the geometry, faces belong only to blocks, and patches as the connectivity between blocks.

Step 4 Merge Block Options

Graphics that illustrate the merge block features will help the user speed up his/her understanding of what kind of superblock grids that can be created. A number of options can be chosen by the user to create superblocks in a desired format. The syntax for including options is:

```
Mrgb file_name [options]
```

Step 5 Limiting Number of Merged Blocks

The user can limit the number of blocks to be merged by using:

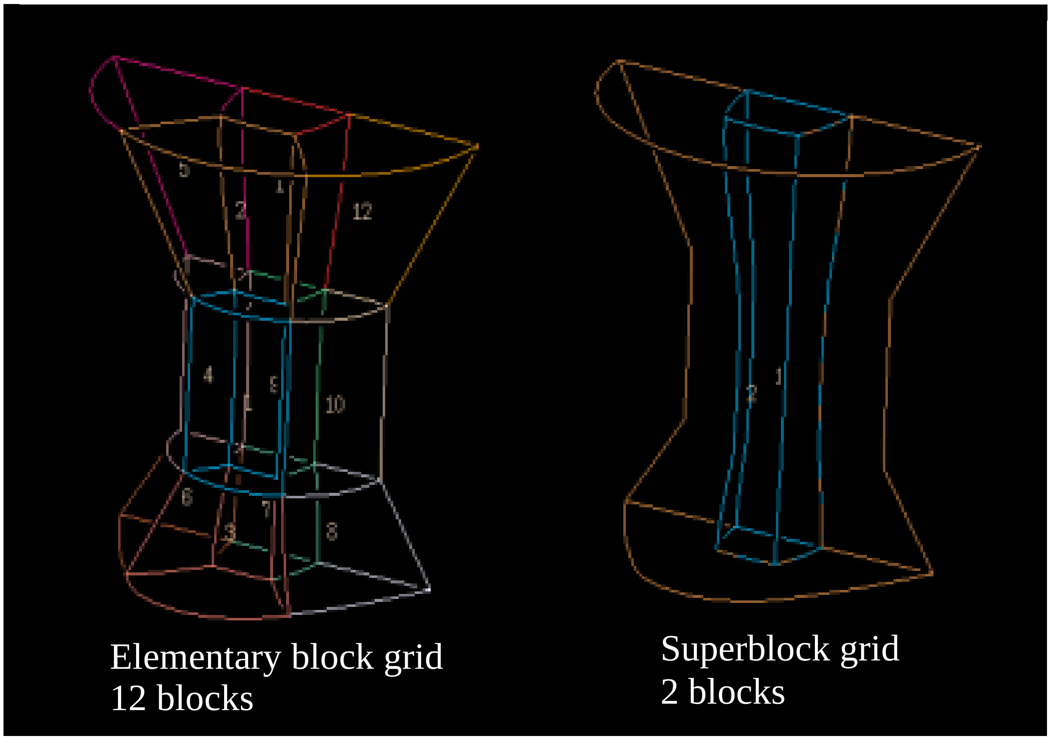
Table 3: Setting Block Number Limits

Option	**Function**
`-maxb [number]`	Limits any merge block to be less than or equal to the number of elementary blocks.
`-maxc [number]`	Limits the merge block to be less than or equal to the number of cells.

Let’s illustrate this option by beginning with a simple grid with 12 elementary blocks. View this grid by loading **nlimit_blk** into the **Grid Viewer panel**. Merge the blocks by using the basic merge block command at the prompt:

```
mrgb Tutorial_8_nlimit_blk <return>
```

The new superblock data will be saved as **`nlimit_blk.tmp`,** load the grid into the **Grid Viewer** panel and compare it with the original elementary block grid. See the pictures below.



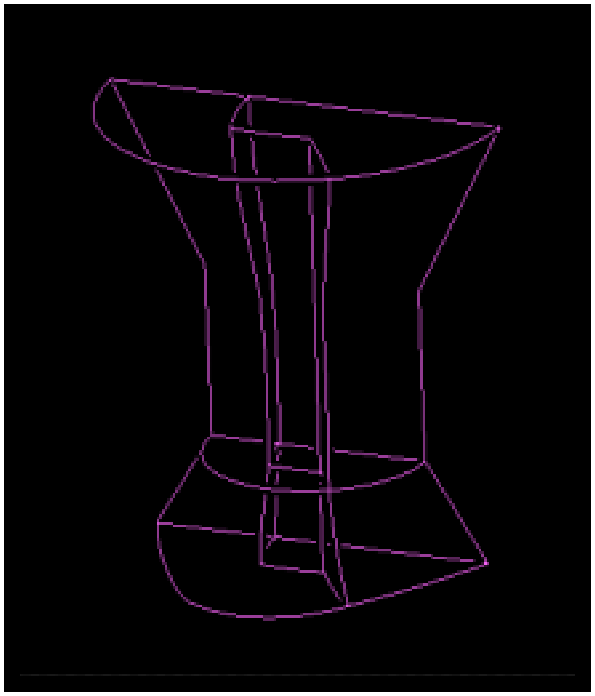
Elementary block grid
12 blocks

Superblock grid
2 blocks

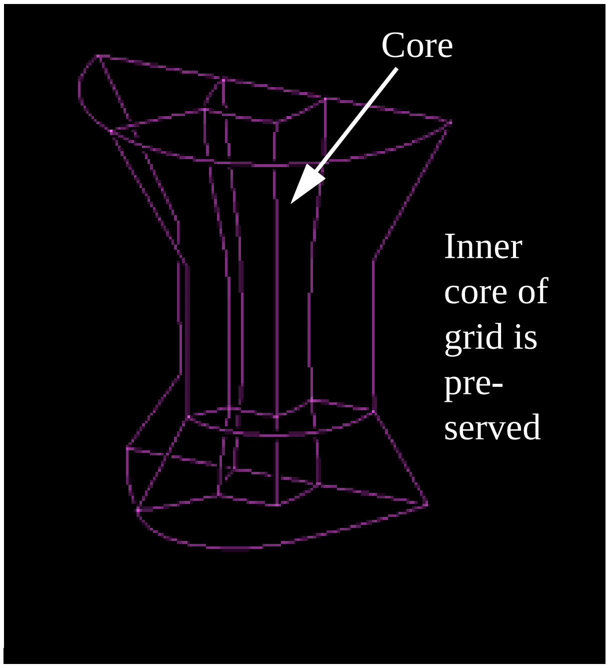
Now let’s increase the number of remaining superblocks from 2 to 4. Go to the prompt and type in:

```
mrgb Tutorial_8_nlimit_blk -maxb 4 <return>
```

The new superblock data will be saved as `nlimit_blk.tmp`. Load the grid into the Grid Viewer and have a look. As you can see, the superblock data has been merged at arbitrary faces along the grid. See the picture below.

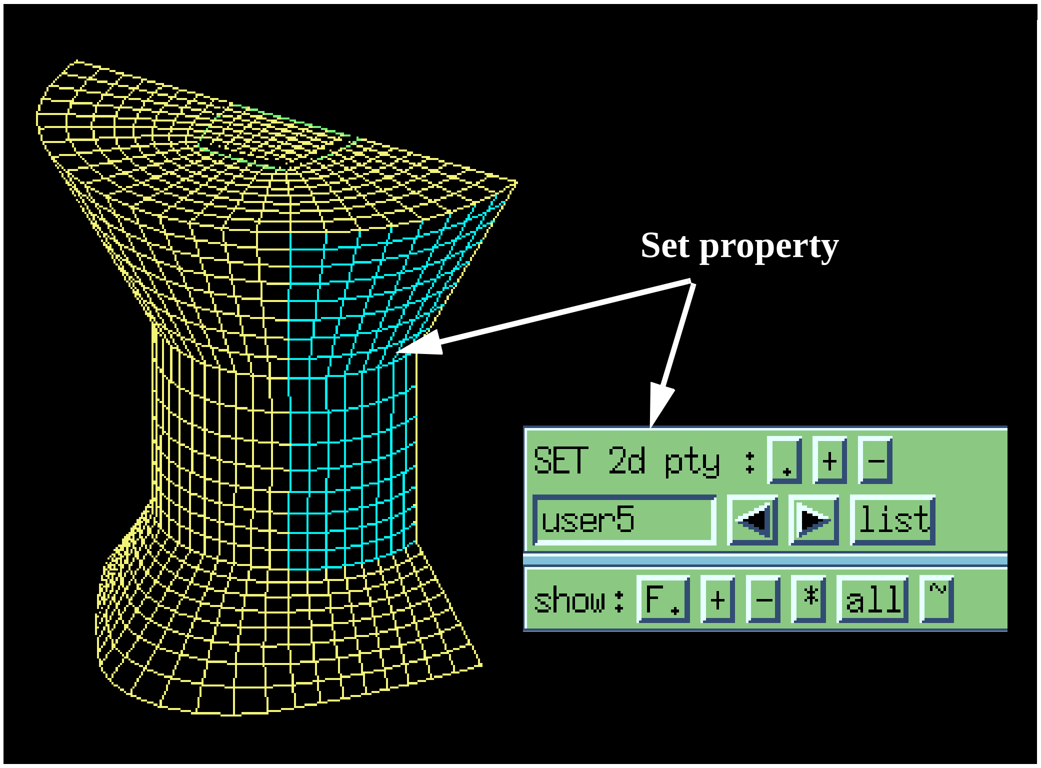


If we set properties along sections of the grid we can control the merging process. Set the core as a **user2** property in **PDC** format, and run the process again. The core will be preserved and a total of four blocks will remain as in the picture below.



Step 6 Merge Block with Face Properties

The user can control the merge block routine by specifying the face properties. Load the grid into the property setter and set two faces on the side as a **PDC user5** property in the **2D Property Setter**. The remaining faces will be set at their default values.



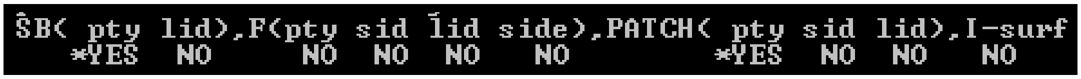
The syntax for whether to consider face property consistency is:

```
mrgb file_name -P <return>
```

Let’s first see what happens if we use the default flag and disregard face property consistency. Go to the prompt and type in:

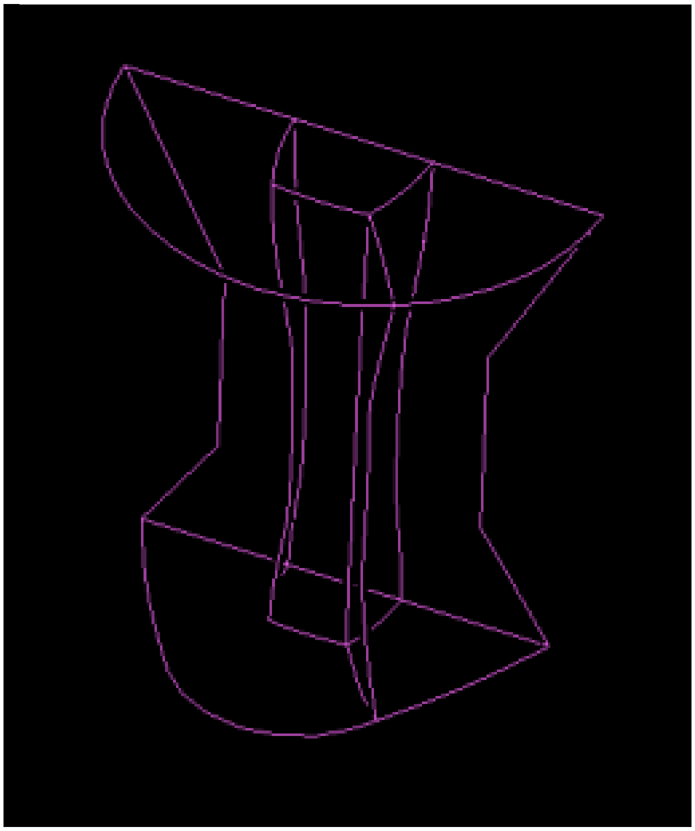
```
mrgb Tutorial_8_facep_blk <return>
```

The result of the merge procedure is shown in the preservation status line that is output to the prompt window, as in the picture below.



```
SB( pty lid),F(pty sid lid side),PATCH( pty sid lid),I-surf
   *YES  NO     NO  NO  NO   NO        *YES  NO  NO   NO
```

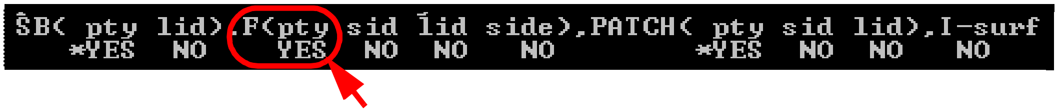
Load the grid into the grid viewer and look at the results.



Now, merge again using the flag:

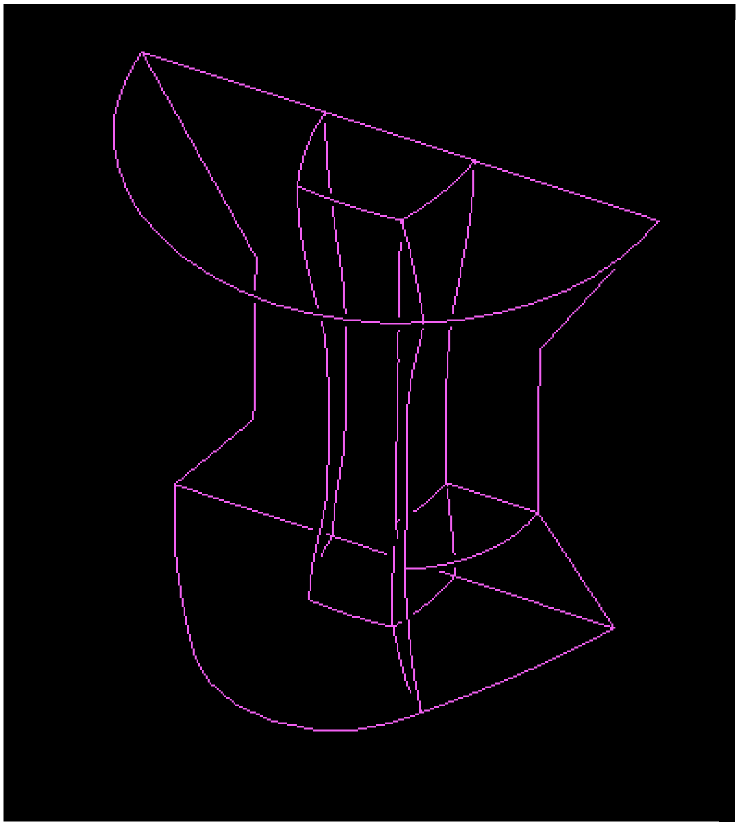
```
mrgb Tutorial_8_facep_blk -P <return>
```

The preservation status line will output a **YES** flag for the face property showing that the superblocks have been made with face property consistency.



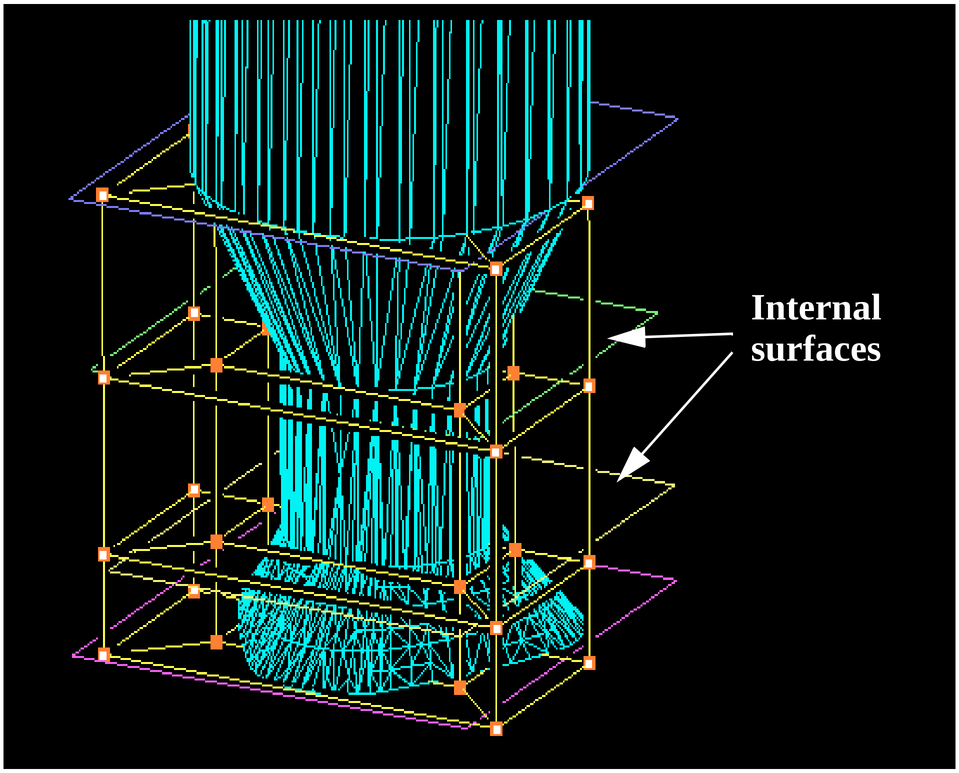
```
ŜB( pty lid),F(pty sid līd side),PATCH( pty sid lid),I-surf
   *YES  NO     YES  NO  NO   NO          *YES  NO  NO   NO
```

Load the grid into the Grid Viewer and look at the results. Notice that the superblocks were not merged across the property settings.



Step 7 Merge using Internal Surfaces

The user can choose whether internal surfaces should be used as a way to restrict the merge process (internal surfaces are those created in the Topology Builder). The internal surfaces used in creating the grid are shown below.



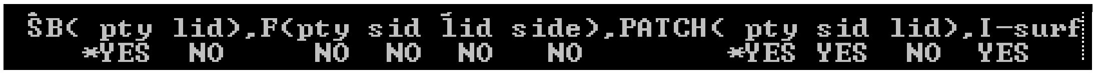
The default merge block setting will merge across internal surfaces, however, if the user does not want to merge through internal surfaces this command can be used:

```
mrgb file_name -I <return>
```

Merge using internal surfaces by going to the prompt and typing in:

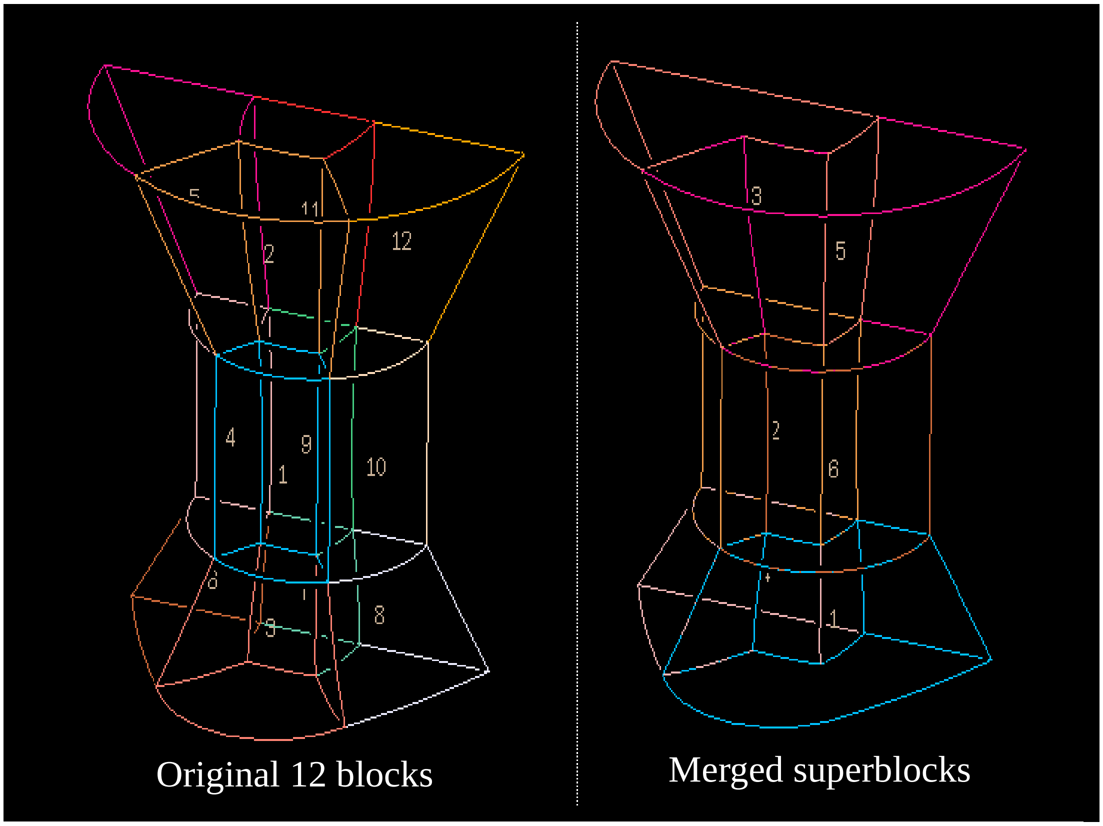
```
mrgb ilimit_blk -I <return>
```

The flags with a YES at the I-surf will appear in the output window:



```
 SB( pty lid),F(pty sid lid side),PATCH( pty sid lid),I-surf
    *YES  NO      NO  NO  NO   NO         *YES YES  NO  YES
```

Load the grid into the grid viewer and look at the results. As you can see, the blocks are not merged across the two internal surfaces



Original 12 blocks

Merged superblocks

Step 8 Scheduled Merge

GridPro allows the user to merge blocks using a predefined routine created in previous runs. As stated above, the schedule merge data is saved in the `file_name.conn_m` file. The command for the scheduled merge is:

```
mrgb file_name_blk -c file_name.conn_m <return>
```

The input and output files and their definition are listed in the table below.

Table 4: Scheduled Merged File Definitions

Input File	**Definition**	**Output File**	**Definition**
`file_name_blk`	Elementary block data	`file_name_blk.tmp`	Merged file in superblock format
`file_name.conn`	Elementary block data connectivity	`file_name_blk.conn_n`	Node connectivity file for the merged blocks
`file_name.conn_m`	Merge schedule file in **GridPro** format		
`file_name.pty`	Property file		

Part III: Merge Block file Descriptions

Property File

Each property file has four sections:

```
24 blocks
#B bid b-pty b-lbid f_i-pty f_i-lbid f_I-pty f_I-lbid ..
B  1  1 -1    1 -1  1 -1  1 -1  1 -1  1 -1  2 -1
B  2  1 -1    1 -1  1 -1  1 -1  1 -1  1 -1  2 -1
B  3  1 -1    1 -1  1 -1  1 -1  1 -1  1 -1  2 -1
B  4  1 -1    1 -1  1 -1  1 -1  1 -1  1 -1  2 -1
B  5  1 -1    1 -1  2 -1  1 -1  1 -1  1 -1  2 -1
B  6  1 -1    1 -1  2 -1  1 -1  1 -1  1 -1  2 -1
B  7  1 -1    1 -1  2 -1  2 -1  1 -1  1 -1  2 -1
B  8  1 -1    1 -1  2 -1  2 -1  1 -1  1 -1  2 -1
B  9  1 -1    1 -1  1 -1  2 -1  1 -1  1 -1  2 -1
B 10  1 -1    1 -1  1 -1  2 -1  1 -1  2 -1  1 -1
B 11  1 -1    1 -1  1 -1  2 -1  1 -1  2 -1  1 -1
B 12  1 -1    1 -1  1 -1  2 -1  1 -1  2 -1  1 -1
B 13  1 -1    1 -1  1 -1  2 -1  1 -1  2 -1  1 -1
B 14  1 -1    1 -1  2 -1  2 -1  1 -1  2 -1  1 -1
B 15  1 -1    1 -1  2 -1  2 -1  1 -1  1 -1  1 -1
B 16  1 -1    2 -1  1 -1  2 -1  1 -1  1 -1  2 -1
B 17  1 -1    1 -1  1 -1  2 -1  1 -1  1 -1  1 -1
B 18  1 -1    1 -1  1 -1  2 -1  1 -1  1 -1  2 -1
B 19  2 -1    1 -1  1 -1  2 -1  1 -1  1 -1  1 -1
B 20  1 -1    1 -1  1 -1  2 -1  1 -1  1 -1  2 -1
B 21  2 -1    1 -1  1 -1  2 -1  1 -1  1 -1  1 -1
B 22  1 -1    1 -1  1 -1  2 -1  1 -1  1 -1  2 -1
B 23  1 -1    2 -1  1 -1  2 -1  1 -1  1 -1  1 -1
B 24  2 -1    1 -1  1 -1  2 -1  1 -1  1 -1  1 -1
 0 labels
 3 32 2D properties (used, max-id) 'PDC    '
  0 unused (pdc:DEFAULT)
  1 interblk (pdc:INTERBLK)
  2 wall (pdc:BOUNDARY)
 3 16 3D properties(used, max-id)
  0 unused (pdc:DEFAULT)
  1 fluid (pdc:BULK)
  2 user2 (pdc:user2)
```

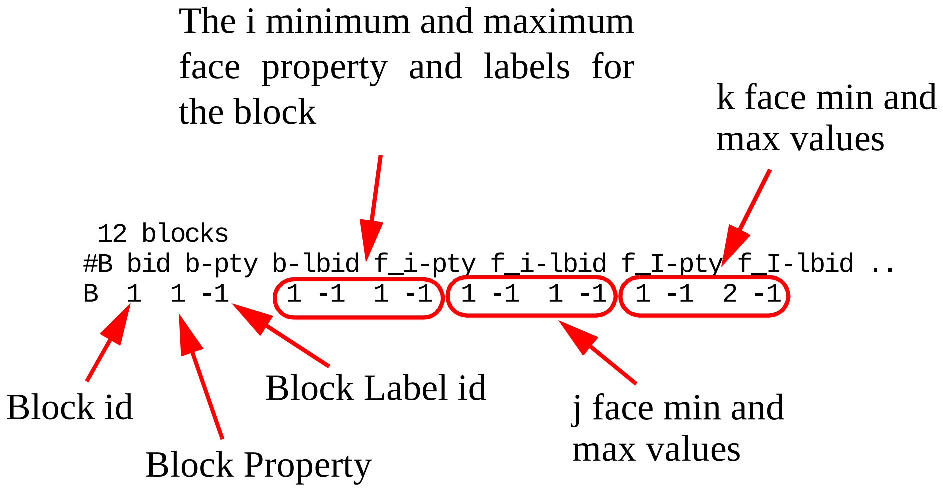
Section 1: Specifies the number of elementary blocks and block properties (see below for detail)

Section 2: Specifies label id’s and string pairs

Section 3: 2D properties and label string pairs

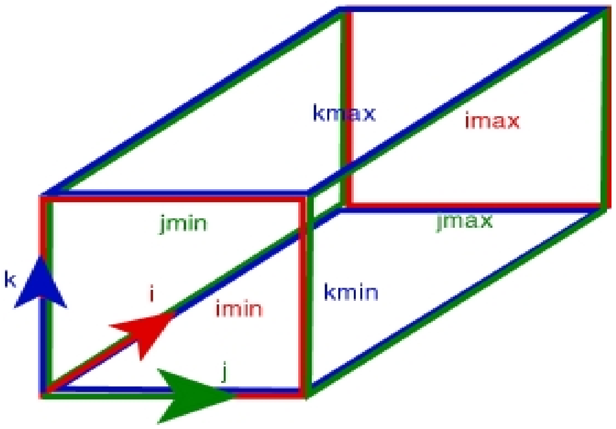
Section 4: 3D properties and label string pairs

Let’s take a look at the block property line in **Section 1** to understand its meaning. Each line begins with a **B** specifying that it is block information and is immediately followed by the block property, label id and the i,j,k values of the property and label for the 6 faces of each block. This is an example of the first line of a 12 block property file:



```
 12 blocks
#B bid b-pty b-lbid f_i-pty f_i-lbid f_I-pty f_I-lbid ..
B  1  1 -1   1 -1  1 -1  1 -1  1 -1  1 -1  2 -1
```

The i minimum and maximum face property and labels for the block
k face min and max values
Block id
Block Property
Block Label id
j face min and max values



The definitions are provided in the table below.

Table 5: Block Property Definitions

Name	**Value**	**Definition**
`bid`	1	First block, the block id is always in sequence.
`b-pty`	1	Block property. The block property definitions are located in section 3. In this case the block property is set at 1 which means that it is interblock data. If it was set at 2, it would be a wall
`b-lbid`	-1	The block label id determined by the user. In this case no label id was specified so it was set at the default value of -1
`f_i-pty`	1	Property value to the face at the minimum i direction. In this case it is set at 1 specifying that it is an interblock.
`f_i-lbid`	-1	Label id for the face in the at the minimum i direction. In this case it is set at the default value of -1
`f_I-pty`	1	Property value to the face at the maximum i direction.
`f_I-lbid`	-1	Label id for the face in the at the maximum i direction
`f_j-pty`	1	Property value to the face at the minimum j direction. In this case it is set at 1 specifying that it is an interblock.
`f_j-lbid`	-1	Label id for the face in the at the minimum j direction.
`f_J-pty`	1	Property value to the face at the maximum j direction.

Table 5: Block Property Definitions

`f_J-lbid`	-1	Label id for the face in the at the maximum j direction
`f_k-pty`	1	Property value to the face at the minimum k direction.
`f_k-lbid`	-1	Label id for the face in the at the minimum k direction.
`f_K-pty`	2	Property value to the face at the maximum k direction. In this case it is set at 2 specifying that it is a wall (see section #2 of the above property)
`f_K-lbid`	-1	Label id for the face in the at the maximum k direction

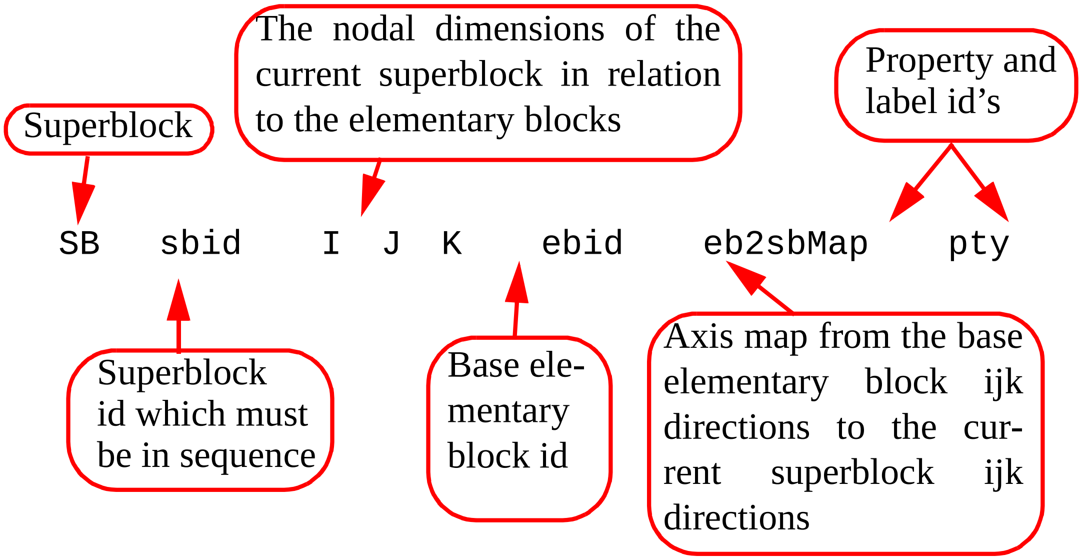
Connectivity Files

The `file_name.conn_m` and `file_name.conn_n` files share the same format. Each file has two sections - one for the merged blocks and one for the face patches. The first line of the two sections give the respective number of merged blocks and patches. Following are the lines defining the connectivity and property of each respective merged block or face patch. An example of a file is shown below.

```
 6  super_blocks
#SB sbid I J K ebid eb2sbMap pty lbid
SB   1   2  3  2    3 012 1 -1
SB   2   2  2  3    4 012 1 -1
SB   3   2  2  3    5 012 1 -1
SB   4   3  2  2    7 213 1 -1
SB   5   2  3  2   12 012 1 -1
SB   6   2  3  2   10 012 1 -1
 29  face_patchs
#P pid sb1 sf1 sb2 sf2 fmap ijk1L 1H  2L 2H pty lbid
P    1   1   5   0   0 012   1 0 0 1 2 1  0 0 0 0 0 0   2 -1
P    2   1   8   0   0 342   0 2 0 1 2 1  0 0 0 0 0 0   2 -1
P    3   1   7   0   0 012   0 0 1 1 2 1  0 0 0 0 0 0   2 -1
P    4   2   8   0   0 012   0 0 0 1 0 2  0 0 0 0 0 0   2 -1
P    5   2   7   0   0 012   0 0 0 1 1 0  0 0 0 0 0 0   2 -1
P    6   3  -3   2   3 012   0 0 0 0 1 2  1 0 0 1 1 2   1 -1
P    7   3   8   0   0 012   0 0 0 1 0 2  0 0 0 0 0 0   2 -1
P    8   3   7   0   0 012   0 0 0 1 1 0  0 0 0 0 0 0   2 -1
P    9   3   4   0   0 012   1 0 0 1 1 2  0 0 0 0 0 0   2 -1

..etc
```

The super blocks have the following format:


Superblock

The nodal dimensions of the current superblock in relation to the elementary blocks

Property and label id's

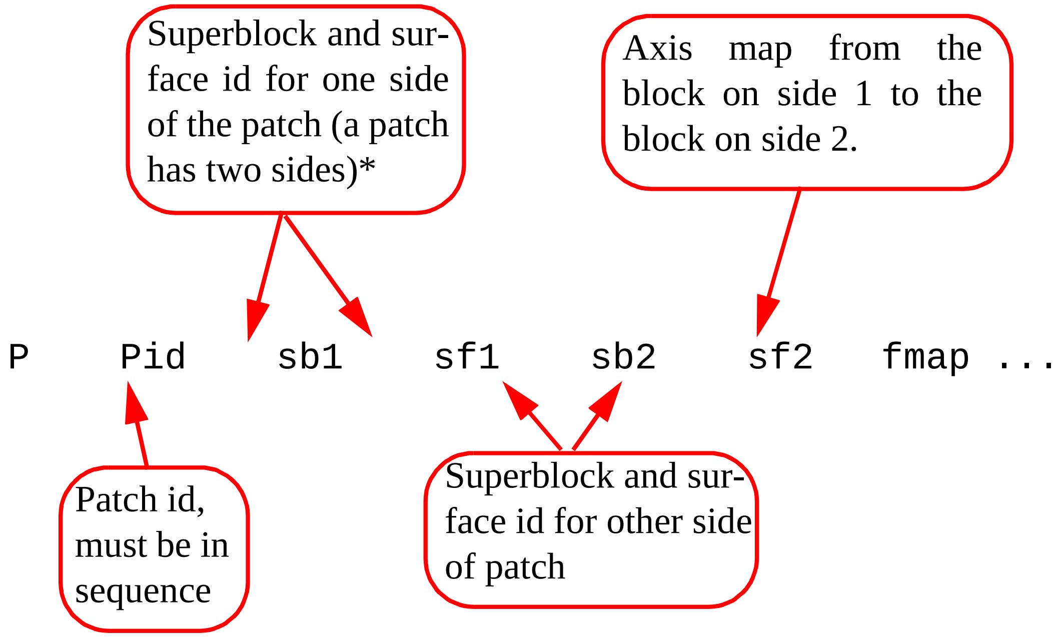
`SB   sbid   I  J  K   ebid   eb2sbMap   pty`

Superblock id which must be in sequence

Base elementary block id

Axis map from the base elementary block ijk directions to the current superblock ijk directions

A patch line in the connectivity file has the following format:



Superblock and surface id for one side of the patch (a patch has two sides)*

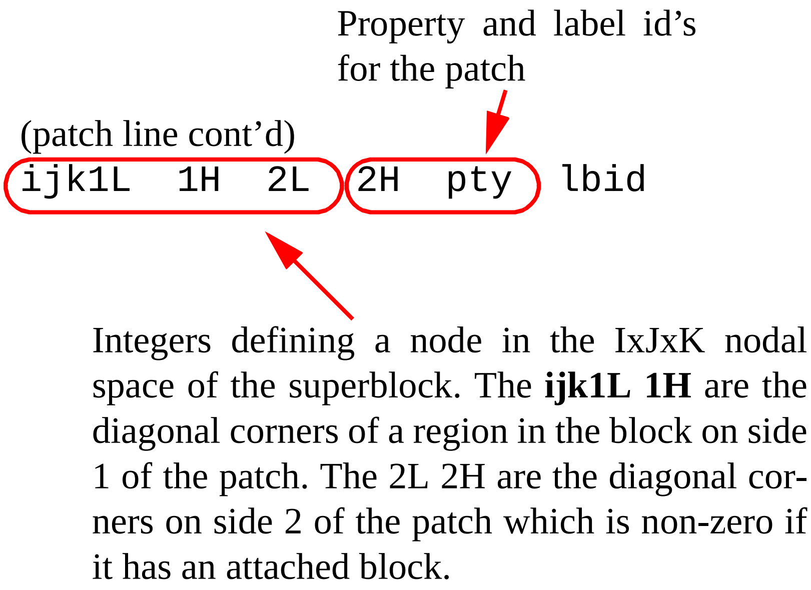
Axis map from the block on side 1 to the block on side 2.

```
P    Pid    sb1    sf1    sb2    sf2   fmap ...
```

Patch id, must be in sequence

Superblock and surface id for other side of patch

*A value of 0 for sb1, sb2, sf1 or sf2 means that no block or surface is attached to that side of the patch.



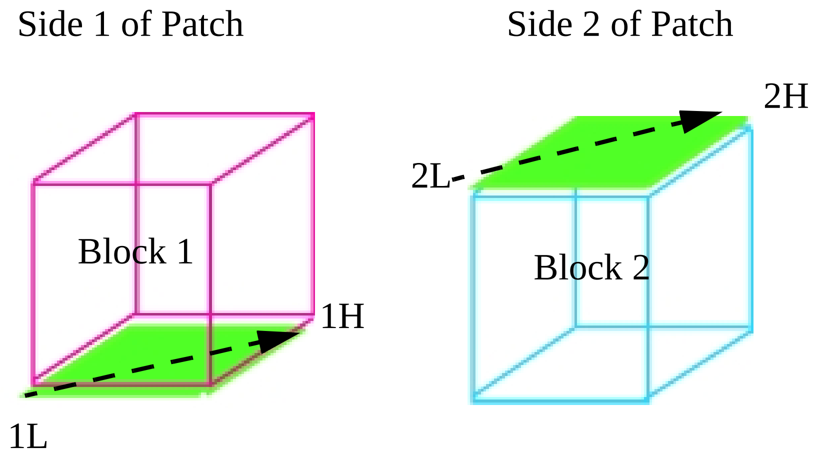
Property and label id's for the patch

(patch line cont'd)

```
ijk1L  1H  2L  2H  pty  lbid
```

Integers defining a node in the IxJxK nodal space of the superblock. The **ijk1L 1H** are the diagonal corners of a region in the block on side 1 of the patch. The 2L 2H are the diagonal corners on side 2 of the patch which is non-zero if it has an attached block.

See illustration below.



For the nodal connectivity **(.conn_n)** file, all of the above applies except that all index related numbers are in terms of cells and cell nodes, and **ebid** and **eb2sbMap** are unused.