FastCap USER'S GUIDE

K. Nabors S. Kim J. White S. Senturia

Research Laboratory of Electronics
Department of Electrical Engineering and Computer Science
Massachusetts Institute of Technology
Cambridge, MA 02139 U.S.A.

18 Sep 1992

This work was supported by the Defense Advanced Research Projects Agency contracts N00014-87-K-825, MDA972-88-K-008, and N00014-91-J-1698, the National Science Foundation contract (MIP-8858764 A02), F.B.I. contract J-FBI-88-067, and grants from I.B.M. Digital Equipment Corporation, and Analog Devices.

Contents

This manual describes FastCap, a three-dimensional capacitance extraction program. FastCap computes self and mutual capacitances between ideal conductors of arbitrary shapes, orientations and sizes. The conductors can be embedded in a dielectric region composed of any number of constant-permittivity regions of any shape and size. The algorithm used in FastCap is an acceleration of the boundary-element technique for solving the integral equation associated with the multiple-conductor, multiple-dielectric capacitance extraction problem. The linear system resulting from the boundary-element discretization is solved using a a generalized conjugate residual algorithm with a fast multipole algorithm to efficiently compute the iterates.

This manual is divided into two sections. The first section explains how to prepare input files for FastCap. The input files contain the description of the conductor surface and dielectric interface geometries. The second section shows how to run the program. It mainly explains how to modify the default settings assumed by FastCap. Also documented in the section are outputs from several example runs, including the postscript files FastCap produces as an aid to problem visualization.

Information on compiling FastCap, obtaining the FastCap source code and corresponding about FastCap is given in Appendix A. Further appendices provide background information about the FastCap algorithm.

1 How to Prepare Input Files

This section of the manual describes how to prepare input files for FastCap. The input files specify the discretization of conductor surfaces into panels. There are two types of input files that can be read by FastCap. One is the PATRAN neutral file, and the other is a generic file format designed for FastCap. These two have unique first line formats, so that FastCap can internally determine the type. Users not wishing to use the PATRAN program to prepare an input file can skip to Section 1.2.

1.1 PATRAN Neutral File Interface

PATRAN is a general purpose, three-dimensional solid modeler with interactive graphics. FastCap includes an interpreter for PATRAN's standard output, the neutral file. This feature allows the user to construct and analyze more complicated conductor structures with help of visualization[1]. This section briefly describes how to use PATRAN to build such solid models, and how to generate the neutral file suitable for FastCap. For a more complete description on using PATRAN, see the PATRAN Release 2.4 user's manual.

1.1.1 PATRAN Data

Before going on any further, one has to understand the hierarchy of data for solid models in PATRAN. There are two levels of data relevant to FastCap, geometry data and analysis model data.

[1] Any geometry that can be read by `fastcap` can also be viewed using the postscript file dump options described in Section 2.5.

Data for geometric entities describe *grids, lines, patches* and *hyperpatches*. This data type is also referred to as phase 1 data in PATRAN. The *grids* are simply points in space. A *patch* is a two-dimensional surface with four corner *grids*, while a *hyperpatch* defines a solid with eight corner *grids*.

Analysis model data, phase 2 data, are built on top of geometric data as they are created by meshing geometric entities. The resulting data consist of *elements* and *nodes*, that are also referred to as CFEG and GFEG data.

1.1.2 Using PATRAN for FastCap

FastCap interprets PATRAN *patches* as conductor surfaces. That is, a conductor is a collection of *patches* that are connected. Discretizing the surfaces into FastCap panels corresponds to meshing the *patches* into PATRAN's quadrilateral and triangular *elements*.

In building conductor models, the following steps should be taken.

1. Build conductor surfaces using *patches* with corner *grids*. All other geometric entities are ignored by FastCap, and can be used only as intermediate entities to help create *patches*. Note: all the patches on a given conductor must be connected. Patches are connected when they share a grid. If this connectivity constraint is ignored, FastCap will interpret the disconnected sets of patches as separate conductors.

2. Name each conductor using the PATRAN NAME command. Each conductor can be named by naming at least one of the conductor's *patches*.

3. Mesh the *patches* using the PATRAN MESH command. This corresponds to placing *nodes* and *elements* on the *patches*. The meshing operation produces a discretization of the conductor surfaces into PATRAN *elements*, also referred to as FastCap panels. One has to remember to create only the zeroth order triangular or quadrilateral *elements*. In PATRAN they correspond TRI/3/0 or QUAD/4/0 *elements*.

4. Finally, write out a neutral file from PATRAN. Be sure to output both geometric (phase 1) and analysis model (phase 2) data to the neutral file (see the PATRAN manual for more information about creating neutral files).

1.1.3 Example

Consider computing the capacitance of the example in Figure 1(a), which consists of two conductors. Each conductor has six faces, and thus requires six *patches* to model their surfaces. Since each set of six *patches* are connected, they will be interpreted as being from one conductor by FastCap. The following is the list of the PATRAN commands to build all the geometric entities in Figure 1(a).

First, put down *grids* to use as *patch* corner points by typing:

```
GRID,  1, , 0.0/0.0/0.0
GRID,  2, , 1.0/0.0/0.0
GRID,  3, , 1.0/1.0/0.0
```

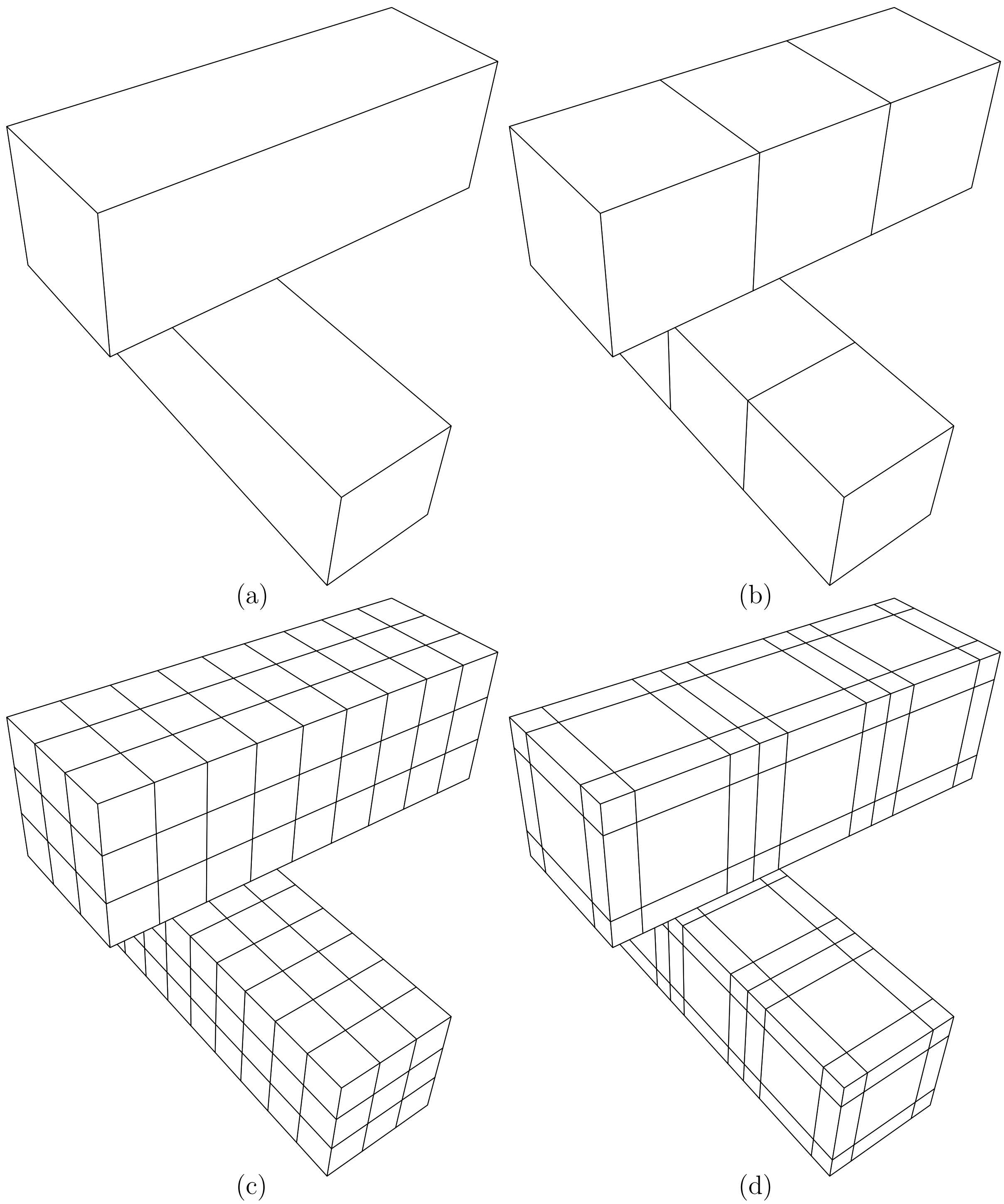

Figure 1: Example Conductor Geometry with Various Discretizations

```
GRID,  4, , 0.0/1.0/0.0
GRID,  5, , 0.0/0.0/3.0
GRID,  6, , 1.0/0.0/3.0
GRID,  7, , 1.0/1.0/3.0
GRID,  8, , 0.0/1.0/3.0
GRID,  9, ,-1.0/2.0/1.0
GRID, 10, ,-1.0/2.0/2.0
GRID, 11, ,-1.0/3.0/2.0
GRID, 12, ,-1.0/3.0/1.0
GRID, 13, , 2.0/2.0/1.0
GRID, 14, , 2.0/2.0/2.0
GRID, 15, , 2.0/3.0/2.0
GRID, 16, , 2.0/3.0/1.0
```

Secondly, construct twelve quadrilateral *patches* by specifying their corner *grid* ID's by typing:

```
PATCH, 1, QUAD, , 1/2/3/4
PATCH, 2, QUAD, , 5/6/7/8
PATCH, 3, QUAD, , 1/4/8/5
PATCH, 4, QUAD, , 4/3/7/8
PATCH, 5, QUAD, , 3/2/6/7
PATCH, 6, QUAD, , 2/1/5/6
PATCH, 7, QUAD, , 9/10/11/12
PATCH, 8, QUAD, , 13/14/15/16
PATCH, 9, QUAD, , 9/12/16/13
PATCH,10, QUAD, , 12/11/15/16
PATCH,11, QUAD, , 11/10/14/15
PATCH,12, QUAD, , 10/9/13/14
```

Then, erase all the geometry entities by deleting from the active set. This can be done by typing:

```
SET, ACTIVE, NONE
```

Plot one *patch* at a time, and use the PATRAN NAME command to name it. Again, only <u>one</u> *patch* from each conductor needs to be named.

```
PATCH, 1, PLOT
NAME, CONDUCTOR1
PATCH, 1, ERASE

PATCH, 7, PLOT
NAME, CONDUCTOR2
PATCH, 7, ERASE
```

Before the next step, plot the entire figure back on the screen by typing:

```
SET, ACTIVE, ALL
```

After the *patches* are created, they must be meshed before creating a FastCap compatible neutral file. As shown in Figure 1(b), PATCH 1 through 12 can be discretized into QUAD/4/0 *elements* by typing:

```
MESH, P1T12, QUAD/4/0, LENGTH, 1.0
```

Begin to write out a neutral file by typing:

```
NEUTRAL
```

PATRAN will produce a couple of neutral system menus. The user should choose CREATE OUTPUT and ENTIRE MODEL options in those menus, respectively. PATRAN will follow with two questions, "Do you wish to output Phase-I data? (Y/N)", and "Do you wish to output GFEG/CFEG tables? (Y/N)". Y responses to both questions will finally generate a neutral file suitable for FastCap.

Now the user is ready to run FastCap with the neutral file by exiting from PATRAN and by typing:

```
fastcap patran.out.1
```

where `patran.out.1` is the name of the neutral file. The user can find more information on how to run FastCap in the next chapter. Shown below is a part of FastCap output that displays the capacitance matrix for the example in Figure 1(b).

```
CAPACITANCE MATRIX, picofarads
                           1         2
CONDUCTOR1%GROUP1 1      135.2    -56.58
CONDUCTOR2%GROUP1 2     -56.58     135.2
```

1.1.4 Modifying the Discretization

In many cases a user might need to generate several discretizations of the conductor surfaces. For example the user might wish to run FastCap with gradually finer meshes to insure that the discretization error is small. The user can rediscretize the surfaces by:

1. Reading in the old PATRAN neutral file, if the user is starting a new PATRAN session.

2. Remeshing the *patches*.

3. Writing out a new neutral file.

As an example, the conductor surfaces in Figure 1(b) can be discretized into smaller panels as shown in Figure 1(c). The following is the list of PATRAN commands to accomplish this panel refinement.

To read in the old neutral file type

```
NEUTRAL
```

PATRAN will respond with a neutral system menu. The user should choose the INPUT MODEL option, and provide the name of the neutral file when asked.

Then, the *patches* can be remeshed by typing:

```
MESH, P1T12, QUAD/4/0, LENGTH, 0.3333
```

PATRAN will respond by asking the user, "WILL YOU PERMIT OVERWRITE? (Y?N?S)" for each patch, since the *patches* are already meshed.

Finally, the user can repeat the last step in the previous section to output a new neutral file.

The following is a part of FastCap output from running the example in Figure 1(c).

```
fastcap bus_x_one_fine_mesh.out

CAPACITANCE MATRIX, picofarads
                            1          2
CONDUCTOR1%GROUP1 1     141.8     -60.81
CONDUCTOR2%GROUP1 2    -60.81      141.8
```

Notice that the capacitance values are different from those reported for the coarse discretization shown in Figure 1(b). The difference is due to the reduced discretization error, which the user must always be aware of when discretizing conductor surfaces.

PATRAN also provides several ways to create non-uniform meshes. The discretization shown in Figure 1(d) is generated in PATRAN by using interactive *node* and *element* editing capabilities. See PATRAN Release 2.4 user's manual for more details.

The results from running FastCap for the discretization is shown below. Note that the values are only slightly different from the previous results in Figure 1(d).

```
fastcap bus_x_one_nonuniform_mesh.out

CAPACITANCE MATRIX, picofarads
                            1          2
CONDUCTOR1%GROUP1 1       143     -61.69
CONDUCTOR2%GROUP1 2    -61.69        143
```

1.2 Generic File Interface

In addition to PATRAN neutral files, FastCap can read files in a simple generic format. Each panel's coordinates are listed in a single line along with its parent conductor name.

The very first line must be a title line, which begins with `0` followed by an arbitrary title as shown below.

```
0  <title>
```

Individual panels representing the discretization of a conductor surface are specified using any mix of two types of data lines, for quadrilateral and triangular panels. Each data line must start with either a `Q` or a `T` followed by the conductor numbers and x, y and z coordinates of four or three corners, respectively, as shown below.

```
Q  <cond. name>  <x1> <y1> <z1>  <x2> <y2> <z2>  <x3> <y3> <z3>  <x4> <y4> <z4>
T  <cond. name>  <x1> <y1> <z1>  <x2> <y2> <z2>  <x3> <y3> <z3>
```

The corner coordinates must be entered in clockwise or counterclockwise order starting from any one corner. The `cond. name` field is usually a number but may be any string in general.

For convenience, each conductor can be renamed using a line which begins with `N`. The line

```
N  <old cond. name>  <new cond. name>
```

associates all the panels previously assigned to `old cond. name` with the new name `new cond. name`.

The following is an example of a generic format input file describing the discretization of one of the two conductors shown in Figure 1(a).

```
Q  1   0.0 0.0 0.0  1.0 0.0 0.0  1.0 1.0 0.0  0.0 1.0 0.0
Q  1   0.0 0.0 0.0  1.0 0.0 0.0  1.0 0.0 1.0  0.0 0.0 1.0
Q  1   0.0 0.0 1.0  1.0 0.0 1.0  1.0 0.0 2.0  0.0 0.0 2.0
Q  1   0.0 0.0 2.0  1.0 0.0 2.0  1.0 0.0 3.0  0.0 0.0 3.0
Q  1   0.0 1.0 0.0  1.0 1.0 0.0  1.0 1.0 1.0  0.0 1.0 1.0
Q  1   0.0 1.0 1.0  1.0 1.0 1.0  1.0 1.0 2.0  0.0 1.0 2.0
Q  1   0.0 1.0 2.0  1.0 1.0 2.0  1.0 1.0 3.0  0.0 1.0 3.0
Q  1   1.0 1.0 0.0  1.0 0.0 0.0  1.0 0.0 1.0  1.0 1.0 1.0
Q  1   1.0 1.0 1.0  1.0 0.0 1.0  1.0 0.0 2.0  1.0 1.0 2.0
Q  1   1.0 1.0 2.0  1.0 0.0 2.0  1.0 0.0 3.0  1.0 1.0 3.0
Q  1   0.0 1.0 0.0  0.0 0.0 0.0  0.0 0.0 1.0  0.0 1.0 1.0
Q  1   0.0 1.0 1.0  0.0 0.0 1.0  0.0 0.0 2.0  0.0 1.0 2.0
Q  1   0.0 1.0 2.0  0.0 0.0 2.0  0.0 0.0 3.0  0.0 1.0 3.0
Q  1   0.0 0.0 3.0  1.0 0.0 3.0  1.0 1.0 3.0  0.0 1.0 3.0
N  1   Big
```

1.3 Using a List File to Specify Complicated Geometries

Both generic and neutral format files may be combined using an auxiliary file called a list file. The list file lists the file names that are to be included in the composite problem and allows for different permittivities to be associated with either side of the surfaces in each of the files. Thus the list file interface provides both a means of entering geometries with multiple dielectrics and, in general, a way to build complicated geometries out of simpler structures.

For example, the two-surface geometry of the coated sphere in Figure 2 can be specified using the list file `coated_sph.lst`,

```
*
* fastcap surface list file for two concentric spherical shells
* - inner is a conductor surface
* - outer is a dielectric/dielectric (coating/air) interface
```

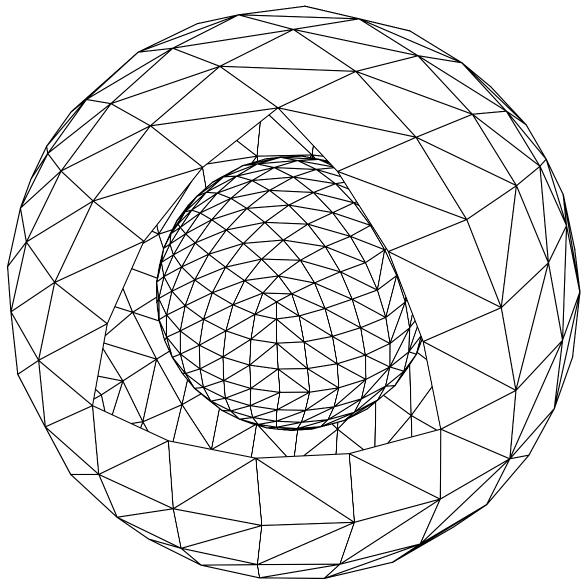

Figure 2: The discretization used to compute the capacitance of a dielectric-coated sphere in free space. Some of the outer dielectric-boundary panels have been removed to show the inner conductor surface panels.

```
*
* radius 1 shell is the conductor
C sphere2.neu 2.0  0.0 0.0 0.0
*
* radius 2 shell is the dielectric interface - air on the outside
D big_sphere1.neu 1.0 2.0  0.0 0.0 0.0  0.0 0.0 0.0 -
```

The first non-comment line specifies the inner, radius-one conductor, surrounded by a dielectric coating with relative permittivity `2.0`. The last line defines the outer surface of the coating, a dielectric interface.

In general a list file has lines of the form given below.

```
* <comment>
G <group name>
C <file> <outperm> <xtran> <ytran> <ztran> [+]
D <file> <outperm> <inperm> <xtran> <ytran> <ztran> <xref> <yref> <zref> [-]
B <file> <outperm> <inperm> <xtran> <ytran> <ztran> <xref> <yref> <zref> [-][+]
```

The `C` line is used to include the panels in `file` as parts of conductor surfaces surrounded by dielectric with relative permittivity `outperm`. The entire contents of `file` may be translated by specifying non-zero values for `<xtran> <ytran> <ztran>`. The `D` and `B` lines specify dielectric interfaces along two regions with permittivities `outperm` and `inperm`. The `B` (or "both") specifies that the dielectric interface is coincident with an infinitesimally thin conductor, while `D` surfaces are dielectric interfaces alone. In both cases, translations are possible as for `C` surfaces. In all cases `file` can be in either PATRAN-generated neutral or generic format.

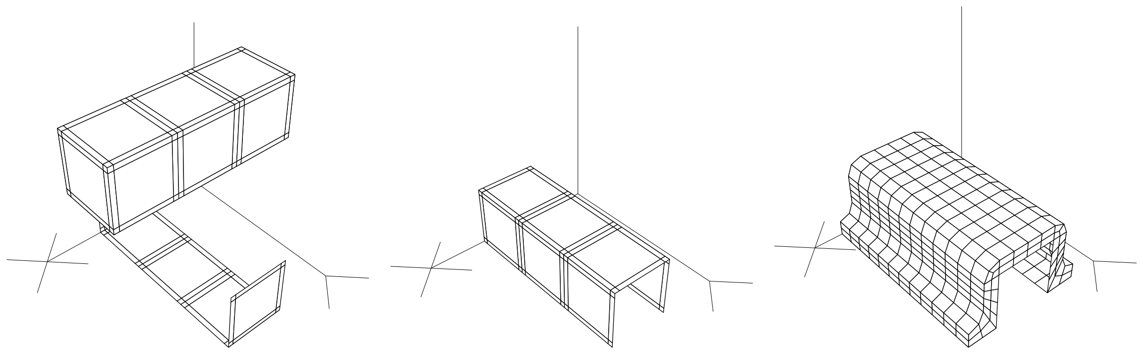

Figure 3: The parts of the 1x1 coated bus crossing problem: conductor-air interfaces, left (from `cond_air_1x1.qui`), conductor-passivation layer interfaces, center (from `cond_dielec_1x1.qui`) and passivation layer-air interfaces, right (from remaining files in `1x1bus.lst`). Axes are three units long.

The reference point `<xref> <yref> <zref>` and optional "`-`" for `D` and `B` surfaces are used to specify which side of the interface has which permittivity. In particular, the reference point `<xref> <yref> <zref>` is assumed to lie on the `outperm` side of all the panels in `file`. The optional "`-`" indicates that `<xref> <yref> <zref>` instead lies on the `inperm` side. It is the user's responsibility to make sure the reference point is on the same side of the dielectric interface for all panels. By partitioning surfaces into convex subsurfaces, a complete specification of any dielectric interface can always be made using multiple files, each with its own reference point.

The optional "`+`" argument is used to control conductor naming. When a `C` or `B` line is terminated with a "`+`," any panels with the same conductor name in that line's `file` and the next `C` or `B` line's `file` are considered to be part of the same conductor. For example, consider the first few lines from the example list file `1x1bus.lst`,

```
*
* 1x1 bus crossing problem with dielectric on lower conductors
*
* conductor to air interfaces
C cond_air_1x1.qui 1.0 0.0 0.0 0.0 +
*
* conductor to dielectric interfaces
C cond_dielec_1x1.qui 7.5 0.0 0.0 0.0
```

The two files `cond_air_1x1.qui` and `cond_dielec_1x1.qui` represent the parts of the conductors that border the air and passivation dielectrics respectively (see Figure 3). Both files contain panels corresponding to the lower conductor with conductor name "`1`." The "`+`" causes all these panels to be grouped together and treated as a single conductor named `1%GROUP1`, as is required for the calculation. Without the "`+`" `fastcap` would associate the "`1`" panels in `cond_air_1x1.qui` with a conductor named `1%GROUP1`, and those in `cond_dielec_1x1.qui` with another conductor named `1%GROUP2`.

If files are in a group (that is they are listed consecutively in the list file and have "`+`" symbols linking them together), all the generic format files must be listed before any PATRAN-generated neutral files. This restriction is necessary due to the ability to rename conductors in generic format files. Also, G lines used to rename groups to be something other than the default GROUP<num>, must occur immediately before the group they rename.

All the features of the list file interface are further illustrated by the example list file test.lst which is generated by typing

```
testgen.sh
```

The script testgen.sh uses cubegen and busgen to create all the input files required by test.lst. Section 2.4.6 describes the geometry corresponding to test.lst.

2 Running FastCap

The basic form of the FastCap program command line is

```
fastcap [-o<expansion order>] [-d<partitioning depth>] [<input file>]
        [-p<permittivity factor>] [-rs<cond list>] [-ri<cond list>]
        [-] [-l<list file>] [-t<iter tol>]
```

Other options having to do with producing pictures of the geometry under analysis are discussed in Section 2.5.

Usually only the input file, in neutral or generic format, or a list file (see Section 1), is specified. For example, the command

```
fastcap sphere2.neu
```

runs fastcap on the example neutral file sphere2.neu described in Section 2.4.1. The input file is taken from stdin if input file is not given. For example

```
busgen | fastcap
```

runs fastcap on the 2×2 bus crossing example discussed in Section 2.4.4. More complicated geometries, in particular problems with multiple dielectric regions, are entered using a list file (see Section 1.3). Thus

```
fastcap -l1x1bus.lst
```

runs fastcap on the coated 1×1 bus crossing problem mentioned in Section 1.3. The "-" option forces a read from stdin in conjunction with a list file or input file read. For example, the capacitance of a cube next to a sphere may be analyzed using

```
cubegen -xo2 | fastcap - sphere2.neu
```

The cubegen generic-format file generator is described in Section 2.4.2. The output is written to stdout except for the postscript files generated using the options described in Section 2.5.

The program automatically chooses the expansion order and spatial partitioning depth that maximize the efficiency of the multipole algorithm and still maintain 1%

accuracy[2] in the larger capacitance matrix entries. By default a uniform, free-space dielectric surrounding the conductors is assumed.

2.1 Trading Off Accuracy, Memory Usage and Speed

The FastCap program's default 1% accuracy is obtained using second order multipole expansions and a dynamically set number of partitioning levels. If lower accuracy is tolerable, faster calculations are possible by forcing `fastcap` to use lower order expansions. For example the command

```
busgen | fastcap -o1
```

runs `fastcap` on the 2×2 bus crossing example (see Section 2.4.4) with first order expansions. The iterates are calculated about twice as fast as for the default case but the capacitance matrix entries will have larger errors, with the greatest percentage error in the smaller matrix entries. Using zero order expansions makes the iterate calculation about nine times faster but can produce significant errors, even in the diagonal capacitance matrix entries, and should not be used when accurate estimate the small coupling capacitances are needed. When using zero order expansions, the partitioning depth should be set manually with the `-d` option, otherwise excessive memory may be used.

In cases where accuracy greater than 1% is required, the tolerance on the iterative loop may be tightened using the `-t` option. The default value `0.01` keeps errors to about 1% of the larger self capacitances. The command

```
busgen -n4 | fastcap -t0.001 -o3
```

results in ten times more accurate solutions to the discretized problem. Third order expansions are specified with the `-o3` option so that the accuracy of the iterates is consistent with the iterative loop tolerance. Also a finer paneling is used (`-n4` option) to bring discretization error down to the level of the tighter tolerance.

2.2 Changing the Dielectric Permittivity Factor

The `-p` option can be used to specify a permittivity factor other than the default value of `1.0`. The permittivity factor multiplies all the permittivities in the problem. The default permittivity is the free-space value 8.8542×10^{-12} F/m but the relative permittivity may be changed[3]. The command line

```
fastcap -p3.5 sphere2.neu
```

runs `fastcap` on the spherical capacitor example embedded in a material with relative permittivity 3.5.

The `-p` option is typically not used with `list file` input since all the permittivities are scaled by the same factor. For example

[2]The linear system arising from the discretization is solved with 1% accuracy. The discretization error must be controlled by the user through a sufficiently fine discretization.

[3]MKSA units are assumed throughout. In particular, conductor dimensions are assumed to be in meters.

```
fastcap -p2.0 -lcoated_sph.lst
```

figures the capacitance of a sphere with a permittivity `4.0` coating embedded in a permittivity `2.0` dielectric medium (see Section 1.3 for a description of `coated_sph.lst`).

2.3 Removing Conductors

The `-rs` and `-ri` options remove conductors from the solve and input lists respectively. The `-rs` option is used to specify a list of conductors that are to be held at zero potential for all the calculations. For example, the problem of a 2×2 bus crossing over a ground plane can be analyzed with the commands

```
busgen > bus.qui
cubegen -p -t -n20 -xo-1 -yo-1 -zo-1 -xh7 -yh7 -naGND \
  | fastcap - bus.qui -rsG
```

The ground plane is modeled using a `cubegen`-generated square sheet with conductor name `GND`. The `-rsG` argument tells `fastcap` to hold the conductor `GND` at zero volts for all the computations. Thus the problem of `GND` at one volt, the four bus wires grounded need not be solved. In general, if the self capacitance of a conductor is not required, it can be removed from the solve list using the `-rs` option. The conductor list argument `<cond list>` is a comma separated list of conductor names of the form

```
[<name>],[<name>],...,[<name>]
```

A `<name>` token need only be enough of the conductor name's leading characters to specify it uniquely.

The `-ri` option is used exactly like `-rs`. All the panels corresponding to conductors in a `-ri <cond list>` are completely removed from the list of input conductors. This option is useful for examining subproblems without altering the input. For example, the capacitance of two wires crossing over a single wire can be examined using

```
busgen | fastcap -ri1
```

2.4 Calculating Capacitances

The output produced by the default version of `fastcap` is described here by way of several examples. As documented in the file `mulGlobal.h`, `fastcap` can be configured to output other information as well.

The first part of the output summarizes the information from the command line and the input file(s). The program then constructs multipole expansions for the potential at all levels of the spatial partitioning. The adaptive nature of the algorithm ensures that only multipole expansions that lead to a reduction in computation are constructed. A message is printed if an entire partitioning level has no multipole or local expansions. With no options specified, `fastcap` automatically sets the partitioning depth, often so that the lowest level is free of expansions. Thus a message of the form

```
No expansions at level 4 (lowest)
```

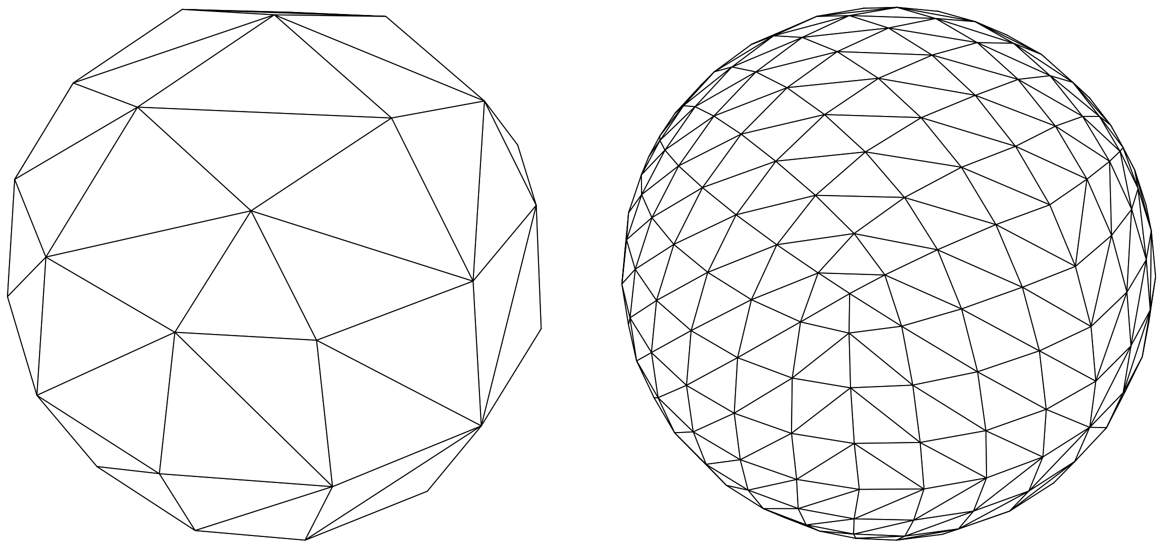

Figure 4: The `sphere1.neu` (left) and `sphere2.neu` discretizations of the unit sphere.

is a normal condition. The final line of multipole setup information has the form

```
Percentage of multiplies done by multipole: 73.1%
```

The number of multiplications done in one `fastcap` iteration for an n panel problem is considerably less than the n^2 required for an explicit matrix-vector product. In this example 73.1% of the n^2 multiplications are approximated using the multipole algorithm, implying a significant reduction in computation.

With the expansions in place, the program begins iterating for the m columns of the capacitance matrix, where m is the number of conductors taking into account those removed from the computation with the `-rs` and `-ri` options. When the iterations are complete, the capacitance matrix entries are printed in MKSA units.

Section 2.4.1 presents example input discretizations of the unit sphere in neutral file format. Other examples are generated by the programs `cubegen` (Section 2.4.2), `capgen` (Section 2.4.3) and `busgen` (Section 2.4.4) in generic file format. More complicated examples, which combine simple geometries generated using the programs `cubegen`, `pipedgen` and `pyragen` with neutral file geometries are given in Sections 2.4.6 through 2.4.8. The generic file generators are compiled with `fastcap` and are all based on the rectangle and triangle discretization functions `disRect()` and `disTri()` in the files `disrect.c` and `distri.c` respectively.

2.4.1 Spherical Capacitor

A conducting sphere in free space with one meter radius has capacitance $4\pi\epsilon_0$ F $\approx$ 0.111265 nF, as may be verified using Gauss's Law. The two example neutral files `sphere1.neu` and `sphere2.neu` describe the two discretizations of the unit sphere illustrated in Figure 4. A third discretization, finer than that in `sphere2.neu`, is given in `sphere3.neu`.

All three discretizations are obtained by meshing a unit sphere primitive with quadrilaterals using the `ISO` option in `PATRAN`. The `sphere1.neu` case corresponds to `ELEM LEN` = `0.5` while the `sphere2.neu` and `sphere3.neu` cases result when `ELEM LEN` = `0.2` and `ELEM LEN` = `0.15` respectively. The panels in the `PATRAN` discretizations are nearly all nonplanar quadrilaterals. Since `fastcap` requires planar panels, each nonplanar quadrilateral in the input file is automatically broken into two planar triangles, resulting in the discretizations of Figure 4.

The command

```
fastcap sphere1.neu
```

produces the output below.

```
Running fastcap 2.0 (25May92)
  Input: sphere1.neu
  Input surfaces:
   GROUP1
    sphere1.neu, conductor
      title: `SPHERE, RADIUS 1, ELEM LEN = 0.5'
      outer permittivity: 1
      number of panels: 104
      number of extra evaluation points: 0
      translation: (0 0 0)
  Date: Mon May 25 15:22:40 1992
  Host: hilbert

INPUT SUMMARY
  Expansion order: 2
  Number of partitioning levels: 3
  Overall permittivity factor: 1
  Total number of panels: 104
    Number of conductor panels: 104
    Number of dielectric interface panels: 0
    Number of thin conductor on dielectric interface panels: 0
  Number of conductors: 1
No expansions at level 3 (lowest)
No expansions at level 2
Percentage of multiplies done by multipole: 0%
Warning: no multipole acceleration

ITERATION DATA
Starting on column 1 (SPHERE%GROUP1)
1 2 3

CAPACITANCE MATRIX, nanofarads
                        1
```

```
SPHERE%GROUP1 1     0.1063
```

This output illustrates how `fastcap` automatically calculates iterates explicitly for problems that are too small to warrant using multipole expansions. For such problems the multipole approximation of iterates is actually more costly than the exact calculation. In this case the coarseness of the discretization also leads to 4% error in the sphere's self-capacitance. Full rated accuracy (default 1%) can only be realized if the discretization error is negligible.

This suggests using a finer discretization of the sphere. The command

```
fastcap sphere2.neu
```

produces the output below.

```
Running fastcap 2.0 (25May92)
  Input: sphere2.neu
  Input surfaces:
   GROUP1
    sphere2.neu, conductor
      title: `SPHERE, RADIUS 1, ELEM LEN = 0.2'
      outer permittivity: 1
      number of panels: 768
      number of extra evaluation points: 0
      translation: (0 0 0)
  Date: Mon May 25 15:24:37 1992
  Host: hilbert

INPUT SUMMARY
  Expansion order: 2
  Number of partitioning levels: 3
  Overall permittivity factor: 1
  Total number of panels: 768
    Number of conductor panels: 768
    Number of dielectric interface panels: 0
    Number of thin conductor on dielectric interface panels: 0
  Number of conductors: 1
No expansions at level 3 (lowest)
Percentage of multiplies done by multipole: 76.1%

ITERATION DATA
Starting on column 1 (SPHERE%GROUP1)
1 2

CAPACITANCE MATRIX, nanofarads
                        1
SPHERE%GROUP1 1     0.1105
```

Compared to the analytic value, the capacitance calculated by `fastcap` now has 0.7% error.

In practice, however, the lack of analytic capacitance values requires further refinement of the discretization until the capacitance values stabilize. The command

```
fastcap sphere3.neu
```

produces the output below.

```
Running ../bin/fastcap 2.0 (25May92)
  Input: sphere3.neu
  Input surfaces:
   GROUP1
    sphere3.neu, conductor
      title: `SPHERE, RADIUS 1, ELEM LEN = 0.15'
      outer permittivity: 1
      number of panels: 1200
      number of extra evaluation points: 0
      translation: (0 0 0)
  Date: Mon May 25 15:32:28 1992
  Host: hilbert

INPUT SUMMARY
  Expansion order: 2
  Number of partitioning levels: 3
  Overall permittivity factor: 1
  Total number of panels: 1200
    Number of conductor panels: 1200
    Number of dielectric interface panels: 0
    Number of thin conductor on dielectric interface panels: 0
  Number of conductors: 1
Percentage of multiplies done by multipole: 77.8%

ITERATION DATA
Starting on column 1 (SPHERE%GROUP1)
1 2

CAPACITANCE MATRIX, nanofarads
                        1
SPHERE%GROUP1 1     0.1108
```

The finer discretization in `sphere3.neu` contains nearly twice as many panels as the `sphere2.neu` case, leading to multipole expansions on the lowest level (level 3).

Compared to the analytic value, the capacitance is correct to better than 1%, as for the `sphere2.neu` discretization. If the analytic capacitance were not known, it would still be possible to conclude that the sphere's capacitance is within 1% of 0.1108nF, since the discretization refinement changes the capacitance only slightly.

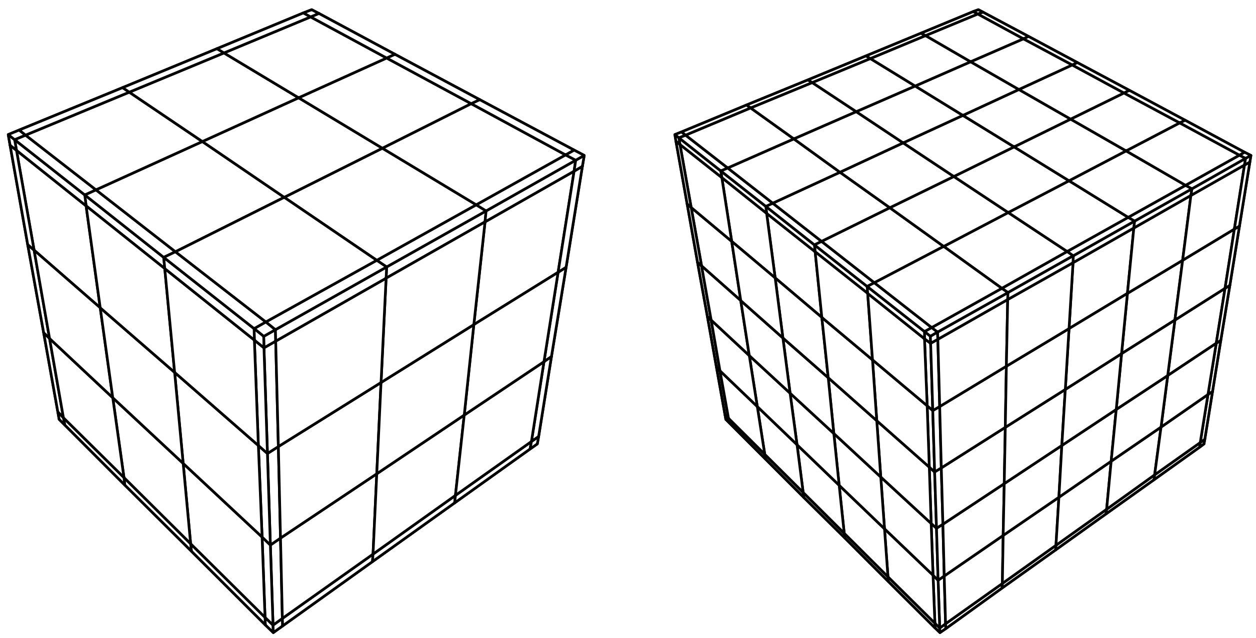

Figure 5: The `cubegen -n5` (left) and `cubegen -n7` discretizations of the unit cube.

Using a discretization refinement to check the capacitance values is not as costly using `fastcap` as for other capacitance extraction programs. Doubling the number of panels when using a Gaussian elimination based algorithm leads to an eight-fold increase in run time. The `fastcap` run time for `sphere3.neu`, however, is only about twice that required for `sphere2.neu`.

2.4.2 Cubic Capacitor

The capacitance of a unit cube in free space may be investigated using the `fastcap` input file generator `cubegen`. The discretization, illustrated on the left in Figure 5, uses five panels along each edge, with edge panel widths equal to 10% of inner panel widths[4]. A `fastcap` generic format input file containing such a discretization is written to `stdout` using the command

```
cubegen -n5
```

The `cubegen` program has several options which are summarized in Section 2.8.

The command

```
cubegen -n5 | fastcap
```

produces the output below.

[4]A. E. Ruehli and P. A. Brennan, "Efficient Capacitance Calculations for Three-Dimensional Multiconductor Systems," *IEEE Transactions on Microwave Theory and Techniques*, 21(2):76–82, February 1973.

```
Running fastcap 2.0 (25May92)
  Input: stdin
  Input surfaces:
   GROUP1
    stdin, conductor
      title: `1mX1mX1m cube (n=5 e=0.1)'
      outer permittivity: 1
      number of panels: 150
      number of extra evaluation points: 0
      translation: (0 0 0)
  Date: Mon May 25 15:37:48 1992
  Host: hilbert

INPUT SUMMARY
  Expansion order: 2
  Number of partitioning levels: 3
  Overall permittivity factor: 1
  Total number of panels: 150
    Number of conductor panels: 150
    Number of dielectric interface panels: 0
    Number of thin conductor on dielectric interface panels: 0
  Number of conductors: 1
No expansions at level 3 (lowest)
Percentage of multiplies done by multipole: 72.6%

ITERATION DATA
Starting on column 1 (1%GROUP1)
1 2 3 4

CAPACITANCE MATRIX, picofarads
                   1
1%GROUP1 1      73.28
```

To check the fidelity of the capacitance value, the finer discretization pictured on the right in Figure 5, with seven panels along each edge, is used. The corresponding command

```
cubegen -n7 | fastcap
```

produces the output below.

```
Running fastcap 2.0 (25May92)
  Input: stdin
  Input surfaces:
   GROUP1
    stdin, conductor
      title: `1mX1mX1m cube (n=7 e=0.1)'
```

```
      outer permittivity: 1
      number of panels: 294
      number of extra evaluation points: 0
      translation: (0 0 0)
  Date: Mon May 25 15:39:15 1992
  Host: hilbert

INPUT SUMMARY
  Expansion order: 2
  Number of partitioning levels: 3
  Overall permittivity factor: 1
  Total number of panels: 294
    Number of conductor panels: 294
    Number of dielectric interface panels: 0
    Number of thin conductor on dielectric interface panels: 0
  Number of conductors: 1
Percentage of multiplies done by multipole: 50%

ITERATION DATA
Starting on column 1 (1%GROUP1)
1 2 3 4 5 6

CAPACITANCE MATRIX, picofarads
                  1
1%GROUP1 1       73.4
```

The small change in the capacitance value indicates that the self-capacitance of the cube is 73.4pF to within 1%. This compares well with other numerical calculations giving values of 73.5pF and 73.4pF[5].

2.4.3 Parallel Plate Capacitor

The capacitance of a parallel plate capacitor can be investigated using the `capgen` input file generator. The command

```
capgen -n20 | fastcap
```

runs `fastcap` on the discretization pictured on the left in Figure 6. The parallel plates are infinitesimally thin, one meter on a side and 0.1 meters apart. Each plate is discretized in the same manner as the cube faces in the previous example, although twenty panels per edge are used to better approximate the stronger singularity in the charge density on a thin plate. The `capgen` command line options are summarized in Section 2.8. The output is given below.

[5]M. A. Jaswon and G. T. Symm, *Integral Equation Methods in Potential Theory and Elastostatics*, Academic Press, London, 1977.

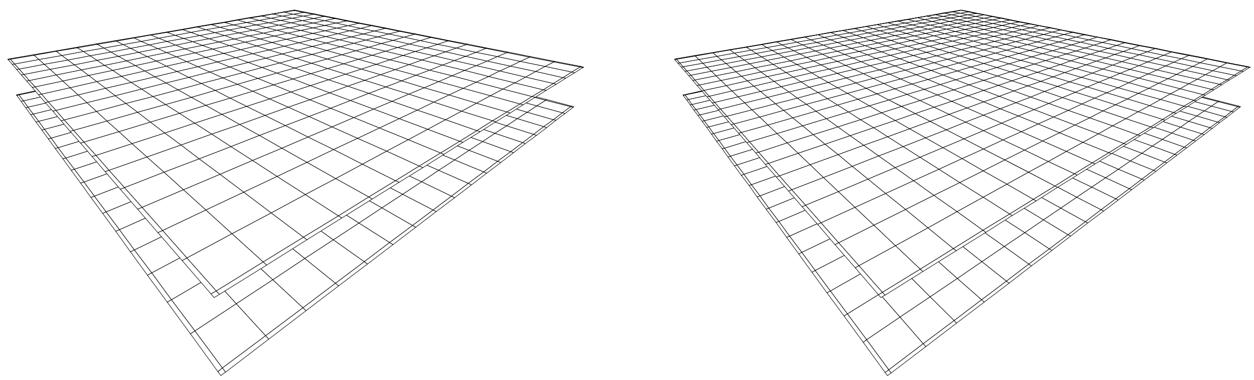

Figure 6: The `capgen -n20` (left) and `capgen -n25` discretizations of a parallel plate capacitor.

```
Running fastcap 2.0 (25May92)
  Input: stdin
  Input surfaces:
   GROUP1
    stdin, conductor
      title: `1mX1m 2 || plate capacitor with 0.1m separation (n=20 e=0.1)'
      outer permittivity: 1
      number of panels: 800
      number of extra evaluation points: 0
      translation: (0 0 0)
  Date: Mon May 25 15:52:45 1992
  Host: hilbert

INPUT SUMMARY
  Expansion order: 2
  Number of partitioning levels: 4
  Overall permittivity factor: 1
  Total number of panels: 800
    Number of conductor panels: 800
    Number of dielectric interface panels: 0
    Number of thin conductor on dielectric interface panels: 0
  Number of conductors: 2
No expansions at level 4 (lowest)
Percentage of multiplies done by multipole: 85.7%

ITERATION DATA
Starting on column 1 (1%GROUP1)
1 2 3
Starting on column 2 (2%GROUP1)
1 2 3 4
```

```
CAPACITANCE MATRIX, nanofarads
                  1          2
1%GROUP1 1     0.1265    -0.1038
2%GROUP1 2    -0.1038     0.1265
```

The finer discretization illustrated on the right in Figure 6 is input to `fastcap` using the command

```
capgen -n25 | fastcap
```

and produces the output below.

```
Running fastcap 2.0 (25May92)
  Input: stdin
  Input surfaces:
   GROUP1
    stdin, conductor
      title: `1mX1m 2 || plate capacitor with 0.1m separation (n=25 e=0.1)'
      outer permittivity: 1
      number of panels: 1250
      number of extra evaluation points: 0
      translation: (0 0 0)
  Date: Mon May 25 15:55:06 1992
  Host: hilbert

INPUT SUMMARY
  Expansion order: 2
  Number of partitioning levels: 4
  Overall permittivity factor: 1
  Total number of panels: 1250
    Number of conductor panels: 1250
    Number of dielectric interface panels: 0
    Number of thin conductor on dielectric interface panels: 0
  Number of conductors: 2
No expansions at level 4 (lowest)
Percentage of multiplies done by multipole: 88.9%

ITERATION DATA
Starting on column 1 (1%GROUP1)
1 2 3 4
Starting on column 2 (2%GROUP1)
1 2 3 4

CAPACITANCE MATRIX, nanofarads
                  1          2
1%GROUP1 1     0.1267     -0.104
2%GROUP1 2     -0.104     0.1267
```

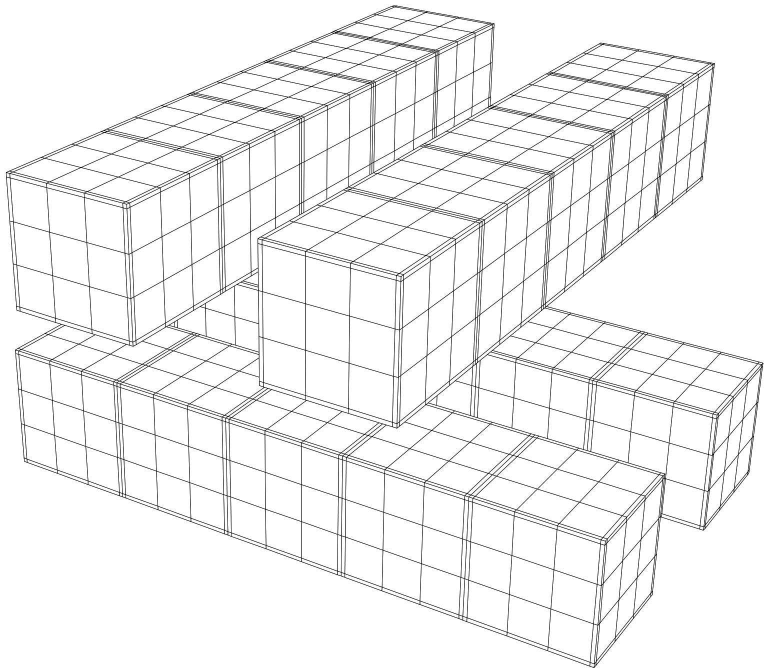

Figure 7: The `busgen -n5` discretization of the 2×2 bus crossing problem.

By the same argument used in previous examples, these capacitances are accurate to 1%.

The entries in the capacitance matrix indicate that when a one volt[6] source is connected across the capacitor, 0.127nC of charge accumulate on the plate at one volt and −0.104nC accumulate on the grounded plate. A small part of the charge on the positive plate, $0.127 - 0.104 = 0.023$nC, is induced to support stray electric field lines terminating at infinity, while the remaining 0.104nC of charge is induced by field lines starting on the positive plate and ending on the grounded plate. If the grounded plate's potential is changed, then an electric field between it and infinity is produced, giving its charge a stray field component. Thus, the most accurate model for the structure in an electric circuit consists of three ideal capacitors: a 0.023nF capacitor from each plate terminal to ground to model stray field effects, and a 0.104nF capacitor between the two terminals. In contrast, the model obtained without `fastcap` is a single capacitor between the terminals with value proportional to the inverse of the plate separation distance, $\epsilon_0/0.1\text{F} = 0.089\text{nF}$. This model completely neglects stray field effects.

2.4.4 Bus Crossing Problem

The final simple example applies `fastcap` to the bus crossing geometry illustrated in Figure 7. The illustrated discretization is generated in generic file format by the program `busgen`, whose options are summarized in Section 2.8. Using a common paneling strategy, `busgen` first breaks the conductors into cubic sections (see Figure 8) and then breaks the exposed faces of each section in the same way as `cubegen` (see Section 2.4.2).

The discretization of Figure 7 is input to `fastcap` with the command

[6] All potentials are referenced to the potential at infinity.

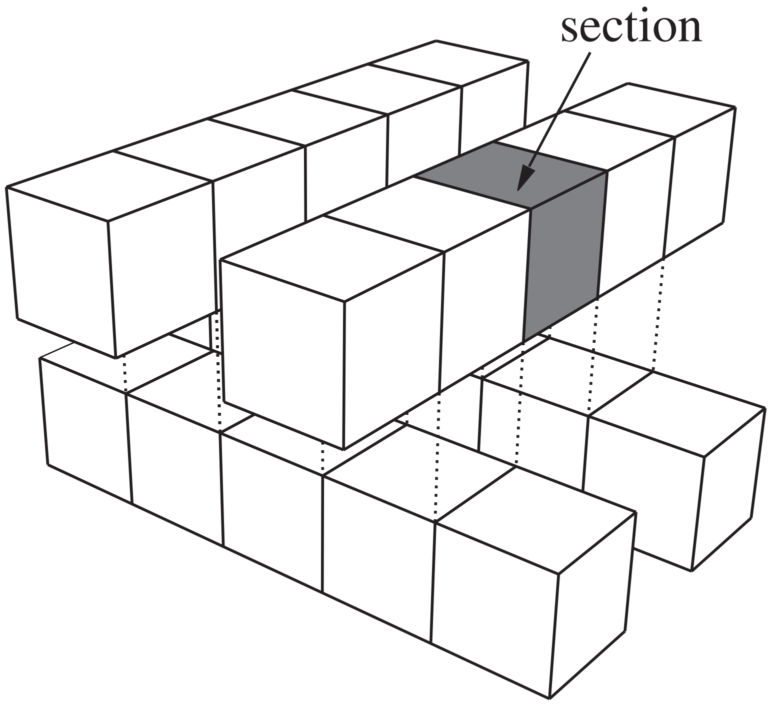

Figure 8: The conductors are broken into sections based on where they overlap.

```
busgen -n5 | fastcap
```

and produces the output below.

```
Running fastcap 2.0 (25May92)
  Input: stdin
  Input surfaces:
   GROUP1
    stdin, conductor
      title: '2X2 bus crossing problem with 1m wires (n=5 e=0.1)'
      outer permittivity: 1
      number of panels: 2200
      number of extra evaluation points: 0
      translation: (0 0 0)
  Date: Mon May 25 15:57:35 1992
  Host: hilbert

INPUT SUMMARY
  Expansion order: 2
  Number of partitioning levels: 4
  Overall permittivity factor: 1
  Total number of panels: 2200
    Number of conductor panels: 2200
    Number of dielectric interface panels: 0
```

```
    Number of thin conductor on dielectric interface panels: 0
  Number of conductors: 4
Percentage of multiplies done by multipole: 89.3%

ITERATION DATA
Starting on column 1 (1%GROUP1)
1 2 3 4 5 6
Starting on column 2 (2%GROUP1)
1 2 3 4 5 6
Starting on column 3 (3%GROUP1)
1 2 3 4 5 6
Starting on column 4 (4%GROUP1)
1 2 3 4 5 6

CAPACITANCE MATRIX, picofarads
                   1          2          3          4
1%GROUP1 1      247.8     -85.01     -48.53     -48.53
2%GROUP1 2     -85.01      247.8     -48.53     -48.54
3%GROUP1 3     -48.53     -48.53      247.9     -84.99
4%GROUP1 4     -48.53     -48.54     -84.99      247.9
```

To check the capacitance values, the discretization of Figure 7 is refined so that each section has seven panels on a side. The new discretization is input to `fastcap` using

```
busgen -n7 | fastcap
```

which gives the output below.

```
Running fastcap 2.0 (25May92)
  Input: stdin
  Input surfaces:
   GROUP1
    stdin, conductor
      title: `2X2 bus crossing problem with 1m wires (n=7 e=0.1)'
      outer permittivity: 1
      number of panels: 4312
      number of extra evaluation points: 0
      translation: (0 0 0)
  Date: Mon May 25 16:05:35 1992
  Host: leibnitz

INPUT SUMMARY
  Expansion order: 2
  Number of partitioning levels: 5
  Overall permittivity factor: 1
  Total number of panels: 4312
```

```
    Number of conductor panels: 4312
    Number of dielectric interface panels: 0
    Number of thin conductor on dielectric interface panels: 0
  Number of conductors: 4
Percentage of multiplies done by multipole: 95.3%

ITERATION DATA
Starting on column 1 (1%GROUP1)
1 2 3 4 5 6 7 8 9 10 11
Starting on column 2 (2%GROUP1)
1 2 3 4 5 6 7 8 9 10 11
Starting on column 3 (3%GROUP1)
1 2 3 4 5 6 7 8 9 10 11
Starting on column 4 (4%GROUP1)
1 2 3 4 5 6 7 8 9 10 11

CAPACITANCE MATRIX, picofarads
                   1          2          3          4
1%GROUP1 1      248.2     -85.18     -48.61     -48.61
2%GROUP1 2     -85.18      248.1     -48.61      -48.6
3%GROUP1 3     -48.61     -48.61      248.3      -85.2
4%GROUP1 4     -48.61      -48.6      -85.2      248.2
```

The small change in the capacitances after the refinement of the discretization indicates that these values are accurate to 1%.

2.4.5 Coated Spherical Capacitor

The list file `coated_sph.lst` described in Section 1.3 represents a unit radius sphere coated with a unit thickness dielectric as illustrated in Figure 2. If the coating has relative permittivity 2 and the surrounding region is free space, the structure has capacitance 148.35pF by Gauss's Law. The command

```
fastcap -lcoated_sph.lst
```

generates the output

```
Running fastcap 2.0 (25May92)
  Input: coated_sph.lst
  Input surfaces:
   GROUP1
    sphere2.neu, conductor
      title: `SPHERE, RADIUS 1, ELEM LEN = 0.2'
      outer permittivity: 2
      number of panels: 768
      number of extra evaluation points: 0
      translation: (0 0 0)
```

```
   GROUP2
    big_sphere1.neu, dielectric interface
      title: ‘SPHERE, RADIUS 2, ELEM LEN = 0.5’
      permittivities: 2 (inner) 1 (outer)
      number of panels: 432
      number of extra evaluation points: 864
      translation: (0 0 0)
  Date: Mon May 25 16:02:13 1992
  Host: hilbert

INPUT SUMMARY
  Expansion order: 2
  Number of partitioning levels: 4
  Overall permittivity factor: 1
  Total number of panels: 1200
    Number of conductor panels: 768
    Number of dielectric interface panels: 432
    Number of thin conductor on dielectric interface panels: 0
  Number of conductors: 1
No expansions at level 4 (lowest)
Percentage of multiplies done by multipole: 90.7%

ITERATION DATA
Starting on column 1 (SPHERE%GROUP1)
1 2 3

CAPACITANCE MATRIX, nanofarads
                        1
SPHERE%GROUP1 1     0.1484
```

The computed capacitance is well within 1% of the analytic value.

2.4.6 A Simple List File Example

A simple example that uses a list file is generated with the command

```
testgen.sh
```

The script `testgen.sh` uses the generic format input file generators `cubegen`, `busgen` and `pyragen` to create input files that are linked together by a list file `test.lst`,

```
*
* test geometry to show how interface works
* - a pyramid, an L-shape and a 1x1 bus xing btwn gnd planes
* - two dielectric slabs between ground planes
*
**   pyramid (single conductor in its own file)
```

```
C ./pyramid.qui 1  0 3 0
**   1x1 bus crossing (single file with multiple conductors)
G BUS
C ./bus.qui 1  0 0 0
**   L-shaped conductor (single object built from 3 separate files)
C ./lcntr.qui 1  0 0 0 +
C ./lleft.qui 1  1 0 0 +
C ./lright.qui 1  0 1 0
**   ground planes (use of same file repeatedly)
C ./plane.qui 1  -1 -1 -1 +
C ./plane.qui 3  -1 -1 5
**   dielectric interface (same file as ground planes)
D ./plane.qui 3 1  -1 -1 4  0 0 5
D ./skirt.qui 1 3  -1 -1 4  0 0 4.5 -
```

The problem consists of a pyramid next to an L-shaped conductor and a 1×1 bus crossing. The three conductors are between two horizontal ground planes. The upper ground plane has a dielectric coating with relative permittivity 3. The geometry is illustrated in Figure 9.

The command

```
fastcap -ltest.lst
```

produces the output

```
Running fastcap 2.0 (25May92)
  Input: ./test.lst
  Input surfaces:
   GROUP1
    pyramid.qui, conductor
      title: `1mX1mX1m pyramid (n=3 e=0.1)'
      outer permittivity: 1
      number of panels: 37
      number of extra evaluation points: 0
      translation: (0 3 0)
   BUS
    bus.qui, conductor
      title: `1X1 bus crossing problem with 1m wires (n=3 e=0.1)'
      outer permittivity: 1
      number of panels: 252
      number of extra evaluation points: 0
      translation: (0 0 0)
   GROUP3
    lcntr.qui, conductor
      title: `1mX1mX1m cube (n=3 e=0.1)'
      outer permittivity: 1
```

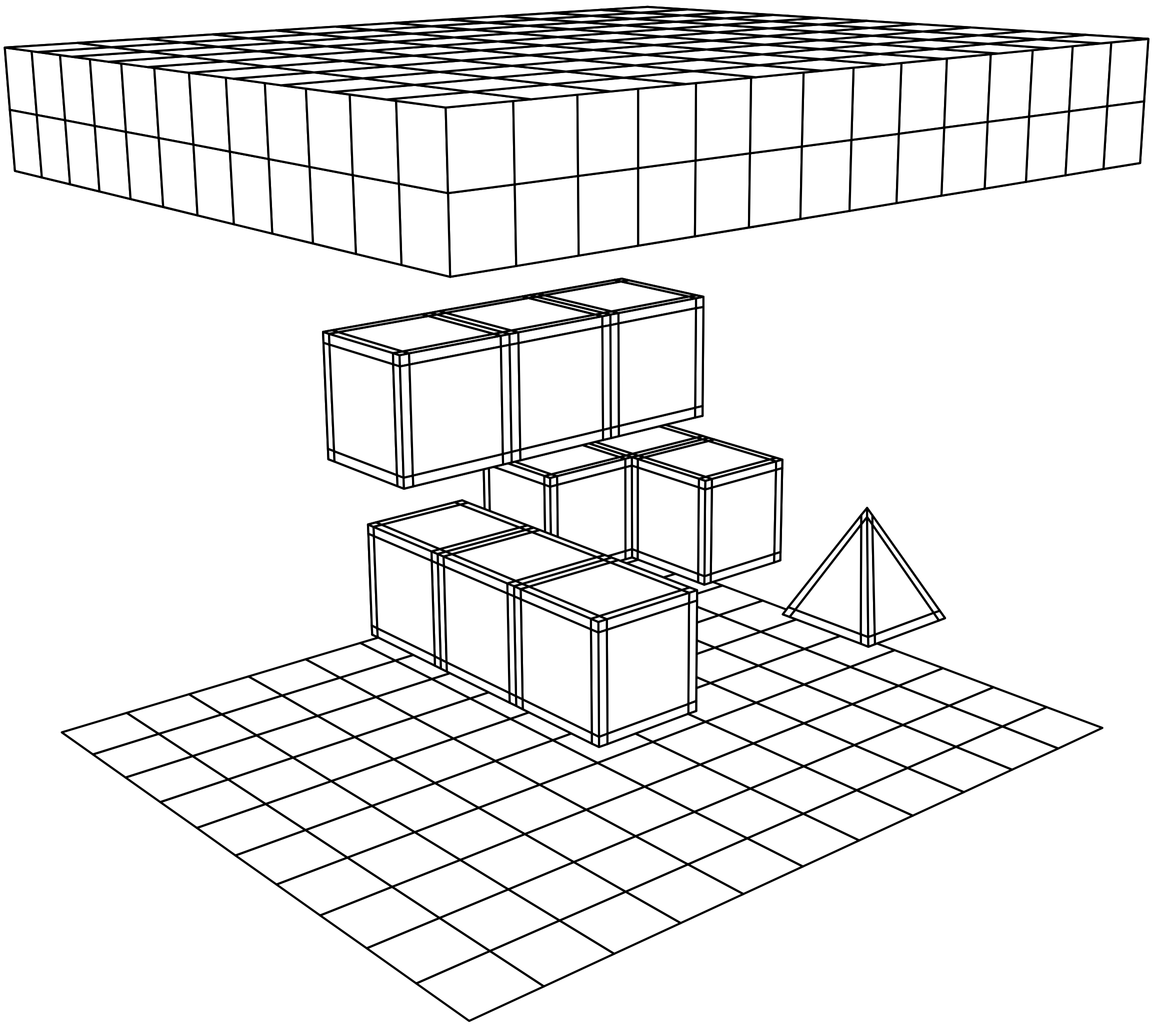

Figure 9: The simple test problem.

```
    number of panels: 36
    number of extra evaluation points: 0
    translation: (0 0 0)
  lleft.qui, conductor
    title: `1mX1mX1m cube (n=3 e=0.1)'
    outer permittivity: 1
    number of panels: 45
    number of extra evaluation points: 0
    translation: (1 0 0)
  lright.qui, conductor
    title: `1mX1mX1m cube (n=3 e=0.1)'
    outer permittivity: 1
    number of panels: 45
    number of extra evaluation points: 0
    translation: (0 1 0)
 GROUP4
  plane.qui, conductor
    title: `7mX6mX1m cube (n=10 e=0)'
    outer permittivity: 1
    number of panels: 110
    number of extra evaluation points: 0
    translation: (-1 -1 -1)
  plane.qui, conductor
    title: `7mX6mX1m cube (n=10 e=0)'
    outer permittivity: 3
    number of panels: 110
    number of extra evaluation points: 0
    translation: (-1 -1 5)
 GROUP5
  plane.qui, dielectric interface
    title: `7mX6mX1m cube (n=10 e=0)'
    permittivities: 1 (inner) 3 (outer)
    number of panels: 110
    number of extra evaluation points: 220
    translation: (-1 -1 4)
 GROUP6
  skirt.qui, dielectric interface
    title: `7mX6mX1m cube (n=2 e=0)'
    permittivities: 3 (inner) 1 (outer)
    number of panels: 104
    number of extra evaluation points: 208
    translation: (-1 -1 4)
Date: Mon May 25 16:38:47 1992
Host: hilbert
```

```
INPUT SUMMARY
  Expansion order: 2
  Number of partitioning levels: 4
  Overall permittivity factor: 1
  Total number of panels: 849
    Number of conductor panels: 635
    Number of dielectric interface panels: 214
    Number of thin conductor on dielectric interface panels: 0
  Number of conductors: 5
No expansions at level 4 (lowest)
Percentage of multiplies done by multipole: 83.2%

ITERATION DATA
Starting on column 1 (PYRAMID%GROUP1)
1 2 3 4 5 6 7 8
Starting on column 2 (1%BUS)
1 2 3 4 5 6 7 8
Starting on column 3 (2%BUS)
1 2 3 4 5 6 7 8 9
Starting on column 4 (LSHAPE%GROUP3)
1 2 3 4 5 6 7 8 9
Starting on column 5 (GND%GROUP4)
1 2 3 4 5 6 7 8 9

CAPACITANCE MATRIX, picofarads
                         1          2          3          4          5
PYRAMID%GROUP1 1     67.54     -4.323      -3.63      -11.3     -43.27
1%BUS          2    -4.323      189.5     -44.44     -17.04       -114
2%BUS          3     -3.63     -44.44      177.3     -11.73     -108.2
LSHAPE%GROUP3  4     -11.3     -17.04     -11.73      171.5     -117.6
GND%GROUP4     5    -43.27       -114     -108.2     -117.6      960.4
```

This example illustrates how the "+" at the end of a list file line can be used to group separated files together. In this case both GROUP3 and GROUP4 are formed this way, producing the conductors LSHAPE%GROUP3 and GND%GROUP4 respectively. In general all panels with the same conductor name in any group are considered to be part of the same conductor. Although this example does not show this, a group may contain more than one conductor.

The group name is always appended to the conductor name read from the input to guarantee that fastcap's internal names are unique. Here the second group has been renamed BUS, rather than the default GROUP2, using the G line in the list file. It is the user's responsibility to rename groups in such a way that all the composite names are unique.

The script testrun.sh gives further examples using this problem.

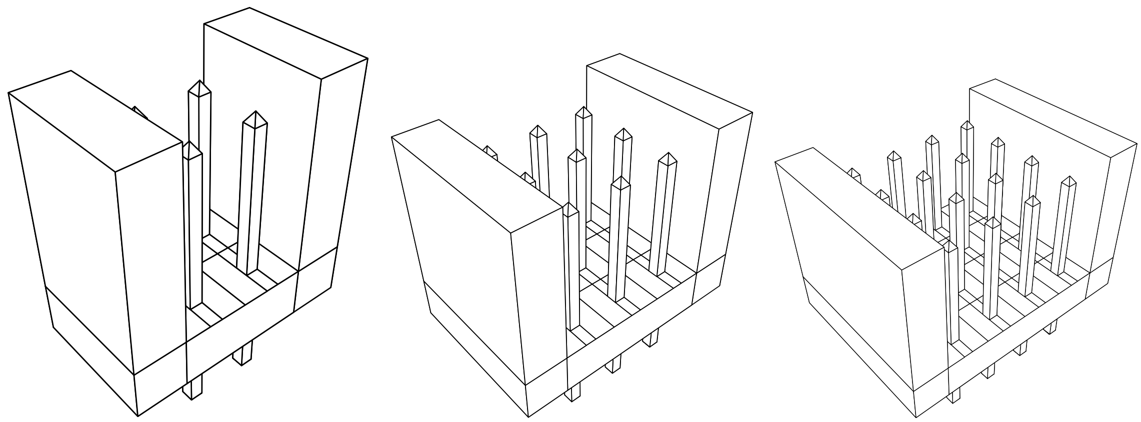

Figure 10: The three backplane connector problems generated by the scripts `connector2.sh` (left), `connector3.sh` (center) and `connector4.sh` (right). The actual discretizations used are finer than those pictured.

2.4.7 Backplane Connector

The capacitances associated with several sizes of backplane conductors may be investigated using the shell scripts `connector2.sh`, `connector3.sh` and `connector4.sh`[7]. Each script generates a set of generic format files using `cubegen` and `pyragen` together with a list file that links the surface files together to produce the final model. The geometries analyzed are pictured in Figure 10. To analyze the smallest connector, first generate the discretization using the command

```
connector2.sh 3.5 -n3
```

The first argument specifies relative permittivity 3.5 for the U-shaped plastic connector body. The second argument is passed directly to `cubegen` and `pyragen` and sets the fineness of the discretization (see also Section 2.8 for more on `cubegen` and `pyragen` options). With the discretization generated, run `fastcap` with the command

```
fastcap -lconnector23.5-n3.lst
```

which gives the output

```
Running fastcap 2.0 (25May92)
  Input: connector23.5-d.lst
  Input surfaces:
   GROUP1
    pin_end10.00.00.03.52-d.qui, conductor
      title: ‘1.3mX1.3mX1m pyramid (n=3 e=0.1)’
      outer permittivity: 1
      number of panels: 4
```

[7]The 4 × 4 conductor connector is discussed in more detail in K. Nabors and J. White, “Multipole-accelerated 3-D capacitance extraction algorithms for structures with conformal dielectrics.” *Proceedings of the 29th ACM/IEEE Design Automation Conference*, June 1992.

```
      number of extra evaluation points: 0
      translation: (0 0 0)
    pin_long0.00.00.03.52-d.qui, conductor
      title: `1.3mX1.3mX15m cube (n=3 e=0.1)'
      outer permittivity: 1
      number of panels: 4
      number of extra evaluation points: 0
      translation: (0 0 0)
    pin_sleeve0.00.00.03.52-d.qui, conductor
      title: `1.3mX1.3mX5.2m cube (n=3 e=0.1)'
      outer permittivity: 3.5
      number of panels: 4
      number of extra evaluation points: 0
      translation: (0 0 0)
    pin_short0.00.00.03.52-d.qui, conductor
      title: `1.3mX1.3mX7m cube (n=3 e=0.1)'
      outer permittivity: 1
      number of panels: 4
      number of extra evaluation points: 0
      translation: (0 0 0)
    pin_end20.00.00.03.52-d.qui, conductor
      title: `1.3mX1.3mX-1m pyramid (n=3 e=0.1)'
      outer permittivity: 1
      number of panels: 4
      number of extra evaluation points: 0
      translation: (0 0 0)
```

(Some surface information removed for brevity.)

```
   GROUP25
    back23.5-d.qui, dielectric interface
      title: `13mX13mX5.2m cube (n=4 e=0)'
      permittivities: 3.5 (inner) 1 (outer)
      number of panels: 2
      number of extra evaluation points: 4
      translation: (0 0 0)
  Date: Tue May 26 01:02:47 1992
  Host: hilbert

INPUT SUMMARY
  Expansion order: 2
  Number of partitioning levels: 3
  Overall permittivity factor: 1
  Total number of panels: 132
    Number of conductor panels: 80
    Number of dielectric interface panels: 52
```

```
    Number of thin conductor on dielectric interface panels: 0
  Number of conductors: 4
No multipole expansions at level 3 (lowest)
Percentage of multiplies done by multipole: 64.9%

ITERATION DATA
Starting on column 1 (1%GROUP1)
1 2 3 4 5
Starting on column 2 (1%GROUP6)
1 2 3 4 5
Starting on column 3 (1%GROUP11)
1 2 3 4 5
Starting on column 4 (1%GROUP16)
1 2 3 4 5

CAPACITANCE MATRIX, nanofarads
                    1          2          3          4
1%GROUP1  1     0.8842    -0.2647    -0.2519    -0.1014
1%GROUP6  2    -0.2647     0.8825    -0.1005    -0.2521
1%GROUP11 3    -0.2519    -0.1005     0.8825     -0.265
1%GROUP16 4    -0.1014    -0.2521     -0.265     0.8818
```

This output further illustrates how `fastcap` groups parts of conductors that are chained together in the list file by means of the `+` option (see Section 1.3). In this particular case, groups 1, 6, 11 and 16, consisting of panels which are all associated with the conductor name "`1`" in the input files, have their corresponding conductors renamed to `1%GROUP1`, `1%GROUP6`, `1%GROUP11` and `1%GROUP16` to avoid any ambiguity.

The input file dimensions are assumed to be in meters, giving capacitances in the nanofarad range. A more realistic connector would have dimensions on the order of millimeters, meaning that the reported capacitances should be interpreted as picofarads.

The example problems for the larger conductors are generated and run in an analogous fashion.

2.4.8 DRAM Cells

The final example uses the shell script `ramgen.sh` to generate a list file that combines generic format files and neutral files to produce a discretization of three adjoining DRAM cells as illustrated in Figure 11[8]. The structure is a simplified model of three adjacent DRAM cells. Each cell consists of a polycide bit line running across the cell, terminating in a via which connects to the drain of a transistor formed by a polycide line crossing the substrate at right angles to the bit line. The source of the transistor is connected

[8]This example is discussed in more detail in K. Nabors and J. White, " An improved approach to including conformal dielectrics in multipole-accelerated three-dimensional capacitance extraction." To appear in *Proceedings of the Workshop on Numerical Modeling of Processes and Devices for Integrated Circuits: NUPAD IV*, May-June 1992.

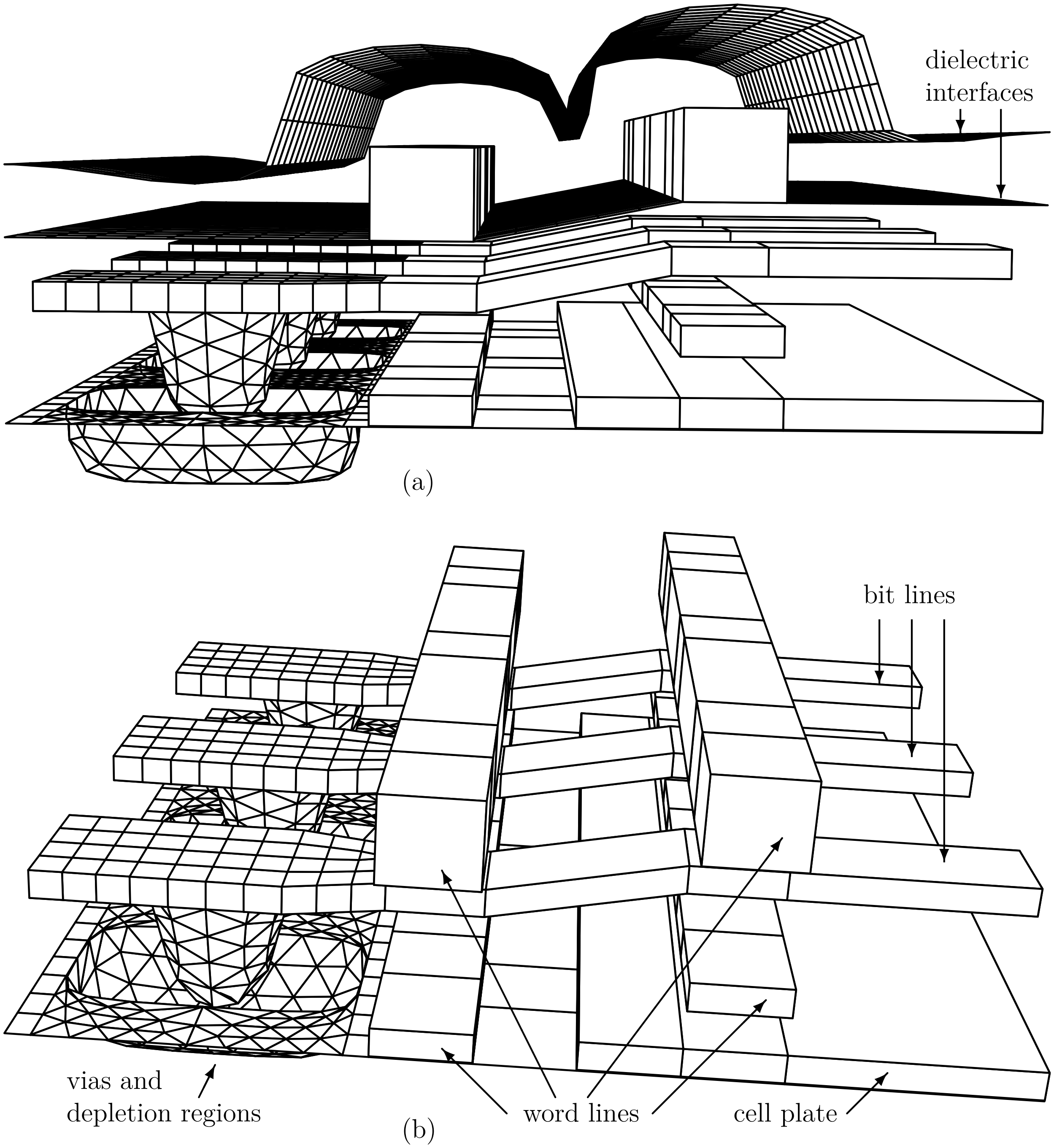

Figure 11: The complete DRAM model, (a), and with the dielectric interfaces removed for clarity, (b).

to a wide polycide plate through a heavily doped region to enhance capacitance. Three more word lines pass through the cells: a polycide line, and two aluminum lines above that line. The aluminum word lines are covered with a silicon nitride (permittivity 7.0) passivation layer, as represented by the two dielectric interfaces illustrated in Figure 11a. The material above the top interface is air while below the aluminum word lines silicon dioxide (permittivity 3.9) is assumed. In Figure 11b, the dielectric interfaces are removed to show the conductors more clearly.

The command

```
ramgen.sh
```

produces a discretization summarized in the list file `ramcell.lst`. The script calls the generic file generator programs `pipedgen` and `cubegen` whose options are itemized in Section 2.8. Typing

```
fastcap -lramcell.lst
```

produces the output below given in the file `ramcell.out`.

As in the connector case the input scale is meters, while the true scale is much smaller. The DRAM cell dimensions are on the order of microns, so the reported values must be scaled by 10^{-6} to obtain the actual capacitances of the DRAM geometry.

2.5 Producing Pictures

The complete command line format of the FastCap program is

```
fastcap [-o<expansion order>] [-d<partitioning depth>] [<input file>]
        [-p<permittivity factor>] [-rs<cond list>] [-ri<cond list>]
        [-] [-l<list file>] [-t<iter tol>] [-a<azimuth>] [-e<elevation>]
        [-r<rotation>] [-h<distance>] [-s<scale>] [-w<linewidth>]
        [-u<upaxis>] [-q<cond list>] [-rc<cond list>] [-x<axeslength>]
        [-b<.figfile>] [-m] [-rk] [-rd] [-dc] [-c] [-v] [-n] [-f] [-g]
```

The first nine basic options are described in previous sections. The remaining options are used to produce postscript format pictures of the geometry under analysis. Section 2.8 itemizes the options and this section gives several examples of their use. The pictures are two-dimensional renderings of the three-dimensional objects under analysis, and may be line drawings or shaded plots indicating the charge density or total charge on each panel. The later plots are useful for gauging the effectiveness of a particular discretization.

2.5.1 Line Drawings

When `fastcap` is given the `-m` option it changes from calculation to line-drawing mode. Usually the default settings produce an acceptable plot for checking a discretization, but many adjustments are possible, the most important being the view angles (relative to a coordinate system parallel to the input coordinates and centered on the center of the object) which are adjusted using the `-a` and `-e` options. Other options control view distance (-h), two-dimensional projection scale (-s), rotation (-r and -u) and line width (-w). Axes of any length may be included with the -x option and lines, arrows and dots may be added using -b. Finally, -rd and -rc are useful for removing dielectric-interface and conductor surfaces, respectively, that block the desired view. Table 1 gives the commands used to produce the line drawings in this guide as examples.

2.5.2 Charge Density and Total Charge Pictures

The -q option provides a means of checking the charge distribution calculated by fastcap. The command

```
fastcap -q sphere1.neu
```

causes fastcap to write the postscript format file sphere11.ps containing the picture in Figure 12. A similar plot can be made for total panel charges using the same command with the -dc option added.

For problems with more than one conductor, fastcap -q writes a postscript file corresponding to the charge density present after each column calculation is completed. The results of any particular subset of the column calculations may be written to postscript files by specifying lists of columns to include with the -q option. For example, in the five conductor problem summarized by the list file test.lst (see Section 2.4.6), fastcap -q2 -ltest.lst produces only one postscript file, the one corresponding to conductor 2%BUS, and fastcap -qPY,LS,2 -ltest.lst writes a picture file for all but conductors 1%BUS and GND%GROUP4.

2.6 Warnings and Errors

The errors and warnings most often encountered when using fastcap are listed here in order of frequency with the most frequent appearing first.

```
Warning: no multipole acceleration
```

The multipole approximation is not being used to compute iterates. As explained in Section 2.4.1, this warning occurs most often when the input problem is too small to warrant the use of the multipole approximation. The message also appears when fastcap is forced to calculate the iterates without the multipole approximation using the -d0 command-line option.

```
resolve_kill_lists: a conductor removed with -ri is in the -rs list
resolve_kill_lists: a conductor removed with -ri or -rs is in the -q list
resolve_kill_lists: all conductors either in -ri or -rs list
```

These errors are caused by asking for plots or computation involving a conductor that has been removed from the problem (with -ri) or is not having its column of the capacitance matrix calculated (in the -rs list).

```
placeq: Warning, removing identical quadrilateral panel
  rmved ctr = (-0.4375 3.43333 1.2) surf = ‘via.neu’ trans = (0 0 0)
  saved ctr = (-0.4375 3.43333 1.2) surf = ‘via.neu’ trans = (0 0 0)
```

Figure	`fasctap` Usage	Comments
3	`fastcap -m -x3 cond_air_1x1.qui`	left figure
	`fastcap -m -x3 cond_dielec_1x1.qui`	center figure
	`fastcap -m -x3 -ltemp.lst`	right figure; `temp.lst` is `1x1bus.lst` with all the "`C`" lines commented out
4	`fastcap -m sphere1.neu`	left figure
	`fastcap -m sphere2.neu`	right figure
5	`cubegen -n5 | fastcap -m`	right figure
	`cubegen -n7 | fastcap -m`	left figure
9	`fastcap -ltest.lst -e75 -m`	
10	`fastcap -m -lconnector23.5-d.lst -s0.5`	left figure; input generated using `connector2.sh 3.5 -d`
	`fastcap -m -lconnector33.5-d.lst -s0.75`	center figure; input generated using `connector3.sh 3.5 -d`
	`fastcap -m -lconnector43.5-d.lst`	left figure; input generated using `connector4.sh 3.5 -d`
11	`fastcap -m -lramcell-d.lst -a0 -e87`	figure (a); input generated using `ramgen.sh -d`
	`fastcap -m -rd -lramcell-d.lst -a0 -e70`	figure (b); input generated using `ramgen.sh -d`
12	`fastcap -q sphere1.neu`	use `-dc` to display total panel charges rather that panel charge densities
13	`cubegen -d | fastcap -m -x2 -bcubeeg.fig`	dots are in `cubeeg.fig`; titling done in latex

Table 1: Commands used to generate some representative figures in this guide.

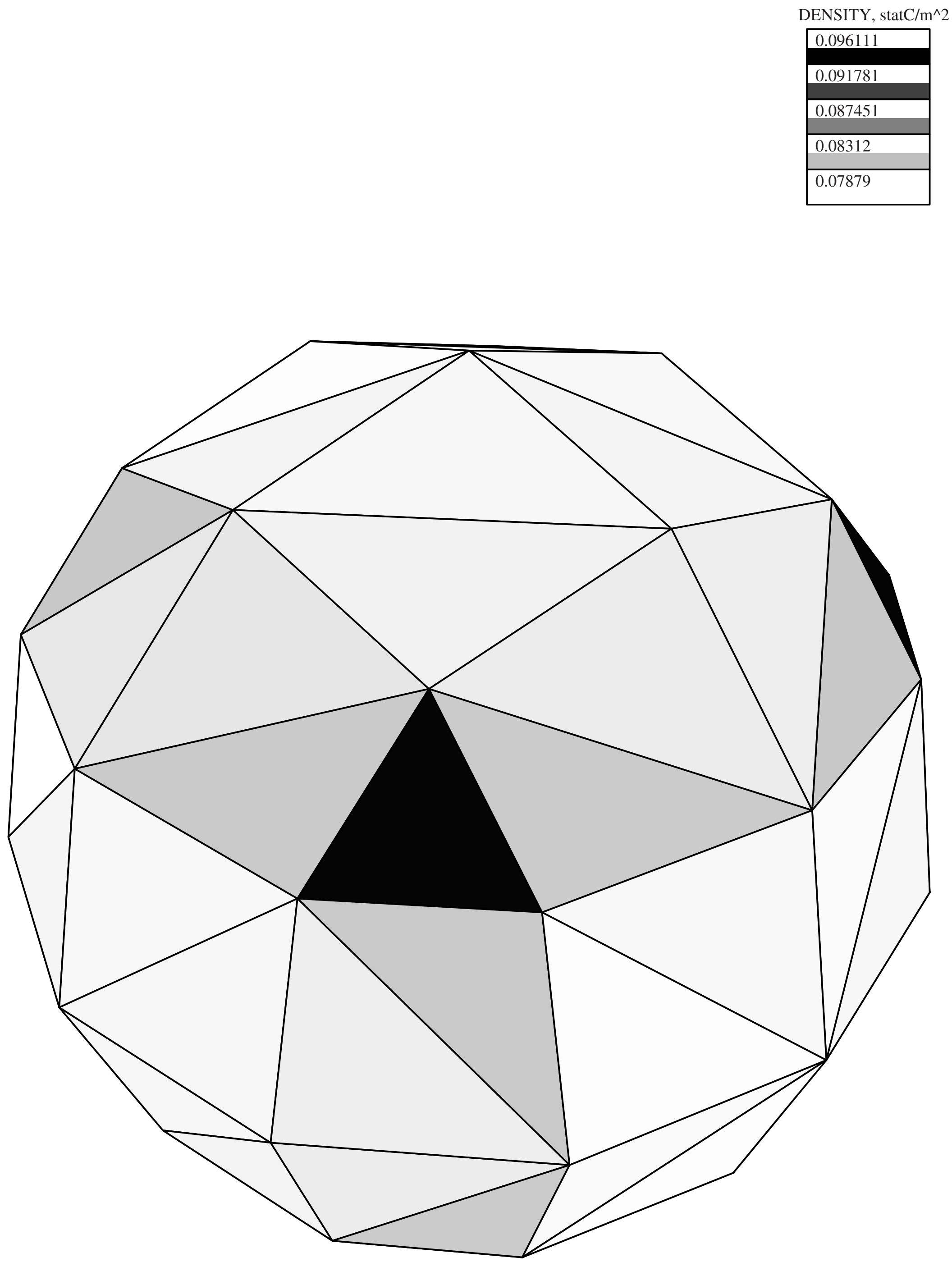

Figure 12: The file `sphere11.ps` resulting from the command `fastcap -q sphere1.neu`, showing the charge density on each panel after the capacitance calculation.

Two panels with the same center coordinates have been given as input. The most common cause of this error is extra panels in neutral files due to careless meshing of patches. The example file `via.neu` contains six duplicate panels. The program removes the extra panels and continues with the calculation.

```
is1stFaceDeeper: Warning, face ordering test failure
  alpha_fac, face 33 = 2.3498 alpha_facref, face 2 = 2.3489
...
```

This error is usually caused by attempting to produce a picture of an illegal discretization with superimposed panels, as can happen when using a list file with incorrect translation vectors. Often the postscript file picture is still correct.

```
read_panels: generic format file
  `../examples/cond_air_1x1.qui'
read after neutral file(s) in same group---reorder list file entries
```

A generic format file occurs after a neutral file in the `list file` and the files are grouped together using the "`+`" option (see Section 1.3). This condition is not allowed due to the conductor renaming that can occur in generic format files.

```
name_data: conductor 'CONDUCTOR NAME'
  has no patch - redo naming so that one is included
assign_names: no conductor names specified
assign_names: 2 conductors have no names
assign_names: 2 names given for 4 conductors
```

The input `PATRAN` neutral file contains conductors that are not named properly. As mentioned in Section 1.1, `fastcap`'s neutral file interface numbers the input conductors in the order they appear in the input file. Since the user normally does not know this order, a component containing at least one patch that is part of each conductor must be named using `PATRAN`'s `NAME` directive.

```
fastcap: out of memory in file `mulSetup.c' at line 135
 (NULL pointer on 1024 byte request)
Total Memory Allocated: 65135 kilobytes (brk = 0x13fff2d8)
placeq: 7 levels set up
```

Virtual memory has been exhausted. As explained in Section 2.1 the usual cause of this problem is running `fastcap` with low order expansions on moderate to large problems. In those cases the number of partitioning levels should be set on the command line with the `-d` option to the amount successfully allocated by the multipole initialization subroutine `placeq`. In this example the option `-d7` should be used. If the error did not occur during the multipole initialization, then the line `placeq: 7 levels set up` is not printed. In that case the number of partitioning levels should be set to one less than that reported in the `INPUT SUMMARY` output section.

```
mksCapDump: Warning - capacitance matrix has non-negative off-diagonals
  row 3 col 1
mksCapDump: Warning - capacitance matrix is not strictly diagonally dominant
  due to row 3
```

The capacitance matrix does not have the correct structure. The warnings are usually encountered when `fastcap` is forced to use lower order expansions. As discussed in Section 2.1, results obtained using lower order expansions are only reliable for the diagonal capacitance-matrix entries. Inaccuracies in the off-diagonal matrix entries can produce these warnings, even though the diagonal capacitances are accurate enough to be useful. The diagonal dominance error is also caused by illegal discretizations with superimposed panels, poorly discretized dielectric regions, particularly with high permittivities, and poorly discretized conductor surfaces, especially when different conductors are very close together.

`NONCONVERGENCE AFTER 2456 ITERATIONS`

The iteration for the capacitance matrix has failed to converge. The default version of `fastcap` halts after n iterations for an n panel problem if convergence has not occurred. This error rarely occurs for practical problems since `fastcap` uses a preconditioner.

2.7 Bugs

This section attempts to itemize the major shortcomings of `fastcap` and the input file generator programs. The user is encouraged to report any other bugs by sending mail to `fastcap-bug@rle-vlsi.mit.edu`.

`fastcap`

1. List file provides for only translations, not rotations.

2. Translations of panels input via `stdin` or `input file` are not possible. Use a list file when translations are required.

3. All input dimensions are assumed in meters. There is currently no way to scale the problem on the command line. The capacitance of the true geometry is equal to the calculated capacitance multiplied by the factor required to convert the input length units (always meters) to the desired units.

4. Dielectric interfaces can only be input with a list file (only conductors can be input through `stdin` and `input file`). Use a list file for all dielectric interfaces.

5. The dielectric surrounding all conductors input through `stdin` and `input file` always has the permittivity of free space multiplied by the permittivity factor specified with `-p` (1.0 by default). Use a list file if other values are required.

6. Dielectric interfaces have no names and therefore cannot be removed selectively from the input or charge distribution plots with command line options. To remove dielectric interfaces selectively from the input, comment out lines in the list file. To remove all dielectric interfaces from charge plots use `-rd`.

7. Infinitesimally thin conductors on dielectric interfaces (included using lines starting with '`B`' in list files), are not supported in this release.

8. All surfaces are listed in the `Input surfaces` part of the output even if all the conductors in some of them have been removed using `-ri`.

9. The PATRAN neutral file interface has very little error checking. If `fastcap` is given an incorrect format neutral file it either hangs or dumps core.

10. The actual postscript filenames used by `fastcap -m` or `fastcap -q<cond list>` are derived from the input file names and cannot be directly changed by the user.

11. Postscript files produced with `fastcap -m` or `fastcap -q<cond list>` produce panel ordering errors (and usually print warning messages) when panels intersect. However, such discretizations are illegal.

12. Occasionally the postscript pictures are incorrect even for legal discretizations. There are two known bugs leading to this problem. The first is often avoided by changing the view angle slightly. The second is caused by problems with dimensions on the order of 10^{-4} or less. This problem can only be avoided by rescaling the input.

13. The panel sorting algorithm used to write out the postscript files takes time proportional to the number of panels squared. This limits its usefulness to problems with less than a few thousand panels. For larger problems the paneling should be done hierarchically so that a coarser discretization can be used for pictures. For problems built using the generic input file generators, the `-d` option provides this function.

14. When axes are included in postscript files using the `-x<axes length>` option, the axes' two-dimensional projections appear in the postscript file before the panel fills. This means that the object should be between the view point and the axes, otherwise the axes can be obscured strangely. Thus the option works best when viewing objects that lie entirely in the positive orthant from a view point in the positive orthant.

15. The bounding box in postscript files produced with `-q` is incorrect if the shading key is included. The bounding box printed is that of the shaded image without the key.

`cubegen`

All the functions of the cube input file generator `cubegen` can be performed by `pipedgen`, although `cubegen`'s interface is often more convenient.

`pyragen`

The pyramid input file generator `pyragen` is currently only capable of the default discretization, with three panels on a side (also results from `-n3`), and no discretization (using `-d`).

`capgen`

All the functions of the parallel plate capacitor input file generator `capgen` can be performed by both `cubegen` and `pipedgen`, unless more than two plates are required.

2.8 Quick Reference

`fastcap`

Run multipole accelerated capacitance extraction algorithm on specified conductor geometry, optionally produce a postscript file picture of the charge distribution used in the calculation, or produce a postscript file line drawing of the input geometry. Arguments can be specified in any order.

```
fastcap [-o<expansion order>] [-d<partitioning depth>] [<input file>]
        [-p<permittivity factor>] [-rs<cond list>] [-ri<cond list>]
        [-] [-l<list file>] [-t<iter tol>] [-a<azimuth>] [-e<elevation>]
        [-r<rotation>] [-h<distance>] [-s<scale>] [-w<linewidth>]
        [-u<upaxis>] [-q<cond list>] [-rc<cond list>] [-x<axeslength>]
        [-b<.figfile>] [-m] [-rk] [-rd] [-dc] [-c] [-v] [-n] [-f] [-g]
```

`fastcap` Options

Option	Default	Range	Function
`-o`	2	`0, 1...`	Specifies expansion order.
`-d`	auto	`0, 1...`	Specifies spatial partitioning depth.
`-p`	`1.0`	$>$ `0.0`	Specifies relative permittivity factor.
`-rs`	‡	‡	Specifies conductors to remove from solve list.
`-ri`	‡	‡	Specifies conductors to remove from input.
`-`	—	—	Forces conductor surface read from `stdin`.
`-l`	—	—	Specifies surface `list file`.
`-t`	`0.01`	$>$ `0.0`	Specifies iterative loop tolerance on $\|residual\|_2$.
`-a`	`50.0`	†	Specifies azimuth view angle in degrees.
`-e`	`50.0`	†	Specifies elevation view angle in degrees.
`-r`	`0.0`	†	Specifies final rotation of 2-D image in degrees.
`-h`	`2.0`	$\geq$ `0.0`	Specifies distance from surface of object in object radii.
`-s`	`1.0`	$>$ `0.0`	Specifies final scaling of 2-D image.
`-w`	`1.0`	$\geq$ `0.0`	Specifies postscript file line width.
`-u`	`z`	`x`, `y`, or `z`	Specifies which 3-D axis is mapped to y-axis in 2-D image.
`-q`	‡	‡	Causes charge distribution postscript files to be written.
`-rc`	‡	‡	Specifies conductors to remove from all pictures.
`-x`	`1.0`	$>$ `0.0`	Includes axes of length `axeslength` in picture.
`-b`	◇	◇	Specifies lines, dots and arrows to superimpose on picture.
`-m`	—	—	Causes postscript-file line drawing write.
`-rk`	—	—	Removes the shading key in charge distribution pictures.
`-rd`	—	—	Removes all dielectric surfaces from postscript file pictures.
`-dc`	—	—	Makes total charge, rather than charge density, `-q` pictures.
`-c`	—	—	Puts the command line in postscript file pictures.
`-v`	—	—	Removes `showpage` from postscript file.
`-n`	—	—	Numbers faces in order input.
`-f`	—	—	Suppresses hidden line removal.
`-g`	—	—	Prints the graph used to order the panels in postscript file.

†Range is unrestricted.
‡Conductor list format: `[<name>],...,[<name>]`. Name strings need only be specified with enough leading characters to be unique. Specifying `-q` with no list is the same as giving a list of all the conductors.
◇ See function `readLines()` in `src/zbufInOut.c` for a description of the `.fig` file format.

`cubegen`

Generate a cubic capacitor problem for input to `fastcap`. Arguments may be specified in any order.

```
cubegen [-xo<originx>] [-yo<originy>] [-zo<originz>]
        [-xh<heightx>] [-yh<heighty>] [-zh<heightz>]
        [-n<num panels/side>] [-e<rel edge panel width>] [-na<name>]
        [-o] [-t] [-b] [-p] [-pfl] [-pfr] [-pbl] [-pbr] [-d]
```

`cubegen` Options

Option	Default	Range	Function
`-xo`	`0.0`	†	Specifies cube origin x coordinate in meters.
`-yo`	`0.0`	†	Specifies cube origin y coordinate in meters.
`-zo`	`0.0`	†	Specifies cube origin z coordinate in meters.
`-xh`	`1.0`	†	Specifies cube excursion from origin in x direction, meters.
`-yh`	`1.0`	†	Specifies cube excursion from origin in y direction, meters.
`-zh`	`1.0`	†	Specifies cube excursion from origin in z direction, meters.
`-n`	`3`	`1, 2...`	Specifies number of panels per cube edge.
`-e`	`0.1`	$\geq$ `0.0`	Specifies edge to inner panel width ratio.
`-na`	'`1`'	‡	Specifies conductor name.
`-o`	—	—	Causes cube to be centered on the point (0 0 0).
`-t`	—	—	Removes panels on the top face, $z =$ `zo + zh` plane◇.
`-b`	—	—	Removes panels on the bottom face, $z =$ `zo` plane◇.
`-p`	—	—	Equivalent to `-pfl -pfr -pbl -pbr`.
`-pfl`	—	—	Removes panels on the front left face, $x =$ `xo + xh` plane◇.
`-pfr`	—	—	Removes panels on the front right face, $y =$ `yo + yh` plane◇.
`-pbl`	—	—	Removes panels on the back left face, $y =$ `yo` plane◇.
`-pbr`	—	—	Removes panels on the back right face, $x =$ `xo` plane◇.
`-d`	—	—	Disables discretization of cube faces.

†Range is unrestricted.
‡Any string with no spaces is allowed.
◇ Face names are based on viewing the cube from the positive orthant. See also Figure 13.

`capgen`

Generate a parallel plate capacitor problem for input to `fastcap`. Arguments may be specified in any order.

```
capgen [-s<plate sep>] [-w<plate width>] [-p<num plates>]
       [-n<num panels/plate width>] [-e<rel edge panel width>]
       [-na<cond name>] [-d]
```

`capgen` Options

Option	Default	Range	Function
`-s`	`0.1`	$>$ `0.0`	Specifies plate separation in meters.
`-w`	`1.0`	$>$ `0.0`	Specifies plate width in meters.
`-p`	`2`	`1, 2...`	Specifies number of parallel plates.
`-n`	`3`	`1, 2...`	Specifies number of panels per plate width.
`-e`	`0.1`	$\geq$ `0.0`	Specifies edge to inner panel width ratio.
`-na`	†	†	Specifies conductor name base.
`-d`	—	—	Disables discretization of plates.

†Any string with no spaces is allowed. By default the conductors are given the first p numbers as names, where p is the number of panels specified with `-p`.

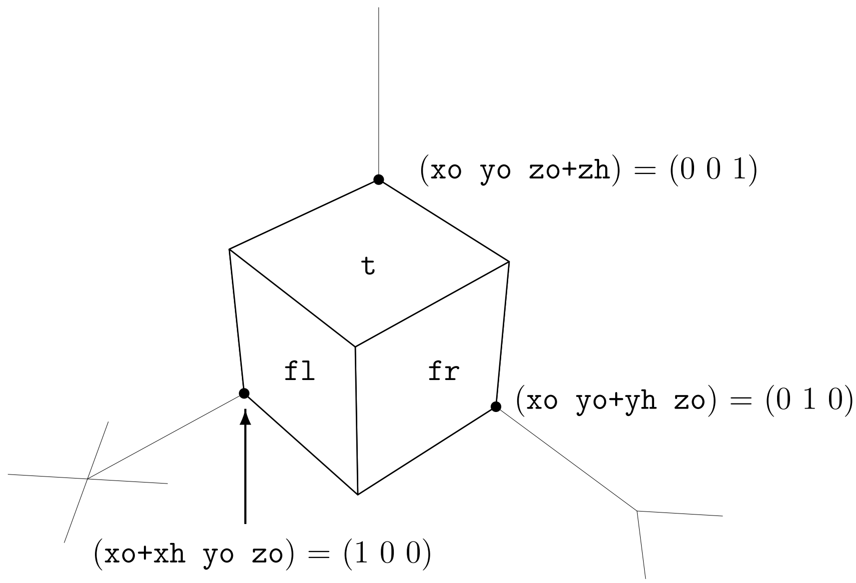

Figure 13: The discretization generated by the command `cubegen -d`, showing how the faces are identified. The axes are two units long.

`busgen`

Generate a bus crossing problem for input to `fastcap`. Arguments may be specified in any order.

```
busgen [-xo<originx>] [-yo<originy>] [-zo<originz>]
       [-c<conductors/bus>] [-w<wire width>]
       [-n<num panels/wire width>] [-e<rel edge panel width>]
       [-na<name base>] [-d]
```

`busgen` Options

Option	Default	Range	Function
`-xo`	`0.0`	†	Specifies bus origin x coordinate in meters.
`-yo`	`0.0`	†	Specifies bus origin y coordinate in meters.
`-zo`	`0.0`	†	Specifies bus origin z coordinate in meters.
`-c`	`2`	`1, 2...`	Specifies number of conductors per bus.
`-w`	`1.0`	$>$ `0.0`	Specifies conductor width in meters.
`-n`	`3`	`1, 2...`	Specifies number of panels per wire width.
`-e`	`0.1`	$\geq$ `0.0`	Specifies edge to inner panel width ratio.
`-na`	‡	‡	Specifies conductor name base.
`-d`	—	—	Disables discretization of conductor faces.

†Range is unrestricted.
‡Any string with no spaces is allowed. By default the conductors are given the first $2c$ numbers as names, where c is the number of conductors per bus specified with `-c`.

`pipedgen`

Generate a parallelepiped problem for input to `fastcap`. Arguments may be specified in any order.

```
pipedgen [-cr <x y z>] [-c1 <x y z>] [-c2 <x y z>] [-c3 <x y z>]
         [-n<num panels/side>] [-e<rel edge panel width>]
         [-na<name>] [-t] [-b] [-p] [-pfl] [-pfr] [-pbl] [-pbr] [-d]
```

`pipedgen` Options

Option	Default	Range	Function
`-cr`	`(0 0 0)`	†	Specifies parallelepiped reference corner.
`-c1`	`(1 0 0)`	†	Specifies first parallelepiped corner.
`-c2`	`(0 1 0)`	†	Specifies second parallelepiped corner.
`-c3`	`(0 0 1)`	†	Specifies third parallelepiped corner.
`-n`	`3`	`1, 2...`	Specifies number of panels per face edge.
`-e`	`0.1`	$\geq$ `0.0`	Specifies edge to inner panel width ratio.
`-na`	‘1’	‡	Specifies conductor name.
`-t`	—	—	Removes panels on the top face, `c1-cr-c2` plane⋄.
`-b`	—	—	Removes panels on the bottom face, parallel to top⋄.
`-p`	—	—	Equivalent to `-pfl -pfr -pbl -pbr`.
`-pfl`	—	—	Removes panels on the front left face, `c1-cr-c3` plane⋄.
`-pfr`	—	—	Removes panels on the front right face, `c2-cr-c3` plane⋄.
`-pbl`	—	—	Removes panels on the back left face, parallel to front right⋄.
`-pbr`	—	—	Removes panels on the back right face, parallel to front left⋄.
`-d`	—	—	Disables discretization of parallelepiped faces.

†Range is unrestricted.
‡Any string with no spaces is allowed.
⋄Face names are based on viewing the parallelepiped from the positive orthant. See also Figure 14.

`pyragen`

Generate a pyramid capacitor problem for input to `fastcap`. Arguments may be specified in any order.

```
pyragen [-xo<originx>] [-yo<originy>] [-zo<originz>]
        [-xh<heightx>] [-yh<heighty>] [-zh<heightz>]
        [-n<num panels/side>] [-e<rel edge panel width>]
        [-na<name>] [-b] [-p] [-pfl] [-pfr] [-pbl] [-pbr] [-d]
```

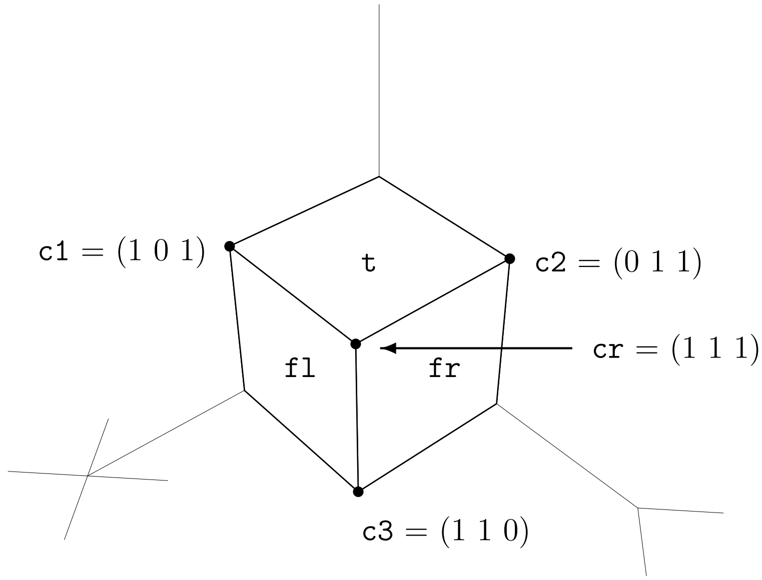

Figure 14: The discretization generated by the command `pipedgen -cr 1 1 1 -c1 1 0 1 -c2 0 1 1 -c3 1 1 0 -d`, showing how the faces are identified. The axes are two units long.

`pyragen` Options

Option	Default	Range	Function
`-xo`	`0.0`	†	Specifies pyramid origin x coordinate in meters.
`-yo`	`0.0`	†	Specifies pyramid origin y coordinate in meters.
`-zo`	`0.0`	†	Specifies pyramid origin z coordinate in meters.
`-xh`	`1.0`	†	Specifies pyramid base excursion from origin in x direction, meters.
`-yh`	`1.0`	†	Specifies pyramid base excursion from origin in y direction, meters.
`-zh`	`1.0`	†	Specifies pyramid height at apex (z direction), meters.
`-n`	`3`	`1, 2...`	Specifies number of panels per cube edge.
`-e`	`0.1`	$\geq$ `0.0`	Specifies edge to inner panel width ratio.
`-na`	‘1’	‡	Specifies conductor name.
`-b`	—	—	Removes panels on the bottom face, $z =$ `zo` plane◇.
`-p`	—	—	Equivalent to `-pfl -pfr -pbl -pbr`.
`-pfl`	—	—	Removes panels on the front left face◇.
`-pfr`	—	—	Removes panels on the front right face◇.
`-pbl`	—	—	Removes panels on the back left face◇.
`-pbr`	—	—	Removes panels on the back right face◇.
`-d`	—	—	Disables discretization of cube faces.

†Range is unrestricted.

‡Any string with no spaces is allowed.

◇ Face names are based on viewing the pyramid from the positive orthant in a manner analogous to the `cubegen` case.

Examples

Run `fastcap` with tighter iterative loop tolerance on `sphere2.neu` with dielectric relative permittivity 3.5:

```
fastcap -p3.5 sphere2.neu -d3 -t0.001
```

Run `fastcap` on a finely discretized $5m \times 5m \times 5m$ cube with zero order expansions for speed and seven partitioning levels to conserve memory:

```
cubegen -n30 -s5.0 | fastcap -o0 -d7
```

Run `fastcap` on the 3×3 bus crossing problem with five uniform panels along short edges:

```
busgen -e1.0 -n5 -c3 | fastcap
```

Run `fastcap` with order zero expansions to get a rough idea of the larger capacitances for a 2μm 4×4 bus crossing embedded in silicon dioxide:

```
busgen -c4 -w2E-6 | fastcap -p3.9 -o0
```

A Compiling FastCap

A tar file containing the source files for `fastcap`, `busgen`, `capgen`, `cubgen` and this guide, as well as all the neutral files it describes, may be obtained on tape by sending a written request to

Prof. Jacob White
Massachusetts Institute of Technology
Department of Electrical Engineering and Computer Science
Room 36-880
Cambridge, MA 02139 U.S.A.

This address may also be used for general correspondence regarding `fastcap`, although electronic mail may be sent to `fastcap-bug@rle-vlsi.mit.edu`, for bug reports, and to `fastcap@rle-vlsi.mit.edu`, for questions or comments, if it is more convenient.

The tar file has the form

```
fastcap-2.0-25May92.tar.Z
```

and yields a two level directory tree when untarred with the commands

```
uncompress fastcap-2.0-25May92.tar.Z
tar xvf fastcap-2.0-25May92.tar
```

The top-level directory is called `fastcap` and has four subdirectories. The directory `fastcap/src` contains all the C source files; the LaTeX files for this manual are found in `fastcap/doc`, and `fastcap/examples` contains the example files. Executables are stored in the `fastcap/bin` directory after compilation using the procedures described below.

A.1 Default Compilation Procedure

The default version of the executable `fastcap` is compiled[9] by changing to the top-level directory, `fastcap`, and typing

```
make all
```

to install the executables `fastcap`, `busgen`, `capgen` and `cubegen` in the `fastcap/bin` directory. Alternatively any of the executables may be compiled alone. For example, "`make fastcap`" produces only `fastcap`.

A.2 Special Compilation Procedures

If `fastcap` is configured to print execution time information by setting the `TIMDAT` flag in `src/mulGlobal.h` to `ON` (see Table 2), an alternate compilation procedure is needed to account for operating system differences. To configure `fastcap/src/Makefile`, change to the top-level directory, `fastcap`, and type

```
config 4
```

for 4.2 or 4.3 BSD systems,

```
config 5
```

for System V systems or

```
config aix
```

for IBM AIX systems. The AIX configuration is recommended for AIX systems even when `fastcap` is not configured to print execution time information. Also certain DEC compilers run out of space during the compile. For those machines use

```
config dec
```

The default configuration may be restored by typing

```
config
```

With `fastcap/src/Makefile` configured, executables are compiled as in the default case.

A.3 Configuration Flags

The file `src/mulGlobal.h` defines flags that configure the discretization, multipole and output routines used in the executable `fastcap`. Table 2 describes the output configuration flags in more detail.

[9]For IBM AIX systems, it is recommended that `fastcap/src/Makefile` be reconfigured before compilation using the procedure described in Section A.2.

Flag	Default		Function
`MKSDAT`	`ON`	`ON`	Prints the symmetrized capacitance matrix.
		`OFF`	Prints nothing.
`CMDDAT`	`ON`	`ON`	Prints the command line arguments.
		`OFF`	Prints nothing.
`RAWDAT`	`OFF`	`ON`	Prints the unsymmetrized capacitance matix.
		`OFF`	Prints nothing.
`ITRDAT`	`OFF`	`ON`	Prints the residual norm after each iteration.
		`OFF`	Prints the iteration number after each iteration.
`TIMDAT`	`OFF`	`ON`	Prints a summary of CPU time and memory usage. Times are only reported if `fastcap` is compiled using the procedure of Section A.2.
		`OFF`	Prints nothing.
`CFGDAT`	`OFF`	`ON`	Prints core configuration flags.
		`OFF`	Prints nothing.
`MULDAT`	`OFF`	`ON`	Prints brief multipole setup information.
		`OFF`	Prints nothing.
`DISSYN`	`OFF`	`ON`	Prints summary of cube involvement by partitioning level.
		`OFF`	Prints nothing.
`DMTCNT`	`OFF`	`ON`	Prints the number of multipole transformation matrices for all possible cube pairings.
		`OFF`	Prints nothing.
`DISSRF`	`ON`	`ON`	Prints surface file information.
		`OFF`	Prints nothing.

Table 2: `fastcap` output configuration compile flags (defined in `fastcap/src/mulGlobal.h`).

A.4 Producing this Guide and Related Documents

In the top-level directory, `fastcap`, type

```
make manual
```

to produce three `.dvi` files `ug.dvi`, `tcad.dvi` and `mtt.dvi` in the directory `fastcap/doc`. The file `ug.dvi` generates the user's guide, while `tcad.dvi` and `mtt.dvi` produce reprints of two articles describing the algorithm used by `fastcap`.