Supplementary users guide to Fastcap (and FFTCap) with Unix hints

Jonathan F. Crawford, Haverford College

July 1999

This supplementary users guide provides instructions for getting started with Fastcap and FFTCap, and includes pointers for those who are less familiar with Unix.

Fastcap and FFTCap are both three dimensional capacitance extraction programs that compute self and mutual capacitances between ideal conductors of arbitrary shapes, sizes and orientations. Fastcap, released in 1992, uses a fast multipole algorithm to solve the integral equations associated with the multiple conductor, multiple dielectric capacitance extraction problem. FFTCap, released in March 1996, instead uses a gridbased, precorrected-FFT algorithm, which in most cases is faster and more memory efficient than the original Fastcap algorithm.

From this point on, “Fastcap” refers to both Fastcap and FFTCap unless noted otherwise.

Fastcap requires a good deal of RAM for complex geometries. There are a number of different steps that can be taken to optimize the performance of Fastcap however, so that acceptable results can be produced with machines with as little as 128 M. Precautions should be taken on machines that have multiple users so as to limit the amount of memory Fastcap is allowed to use, so that other users can run programs too. Under Unix, this can be done with the limit command. For example

```
>limit datasize 92160
```

limits the amount of memory to 92160 blocks of 512-byte blocks, or about 47 Mb. If a particular geometry causes Fastcap to exceed the limit set by the limit command, that run will simply halt. During a particular run, if you wish to monitor the amount of systems resources Fastcap is using, Fastcap has to be run in the background. This can be accomplished by ending the command line with a &. For example

```
>fastcap -lsmallfry22.1st &
```

Now, other commands can be executed while Fastcap is running. The command

```
>top
```

produces a list of current users and the percent of system resources they are using, as shown below:

```
================================================================================

load averages:  0.01,  0.01,  0.01                                      11:18:56
65 processes:  64 sleeping, 1 on cpu
CPU states:     % idle,     % user,     % kernel,     % iowait,     % swap
Memory: 256M real, 63M free, 5432K swap in use, 379M swap free

  PID USERNAME THR PRI NICE  SIZE   RES STATE   TIME    CPU COMMAND
 5877 joseph     1  59    0 1272K  992K cpu     0:00  0.24% top
 3916 wlosert    1  59    0   37M   32M sleep  19:30  0.04% netscape
22058 root       1  58    0 2696K 1104K sleep   0:30  0.02% lpsched
  211 root      10  51    0 3152K 2736K sleep   3:55  0.01% nscd
  230 root       1  58    0   39M   37M sleep  11:22  0.00% lp
  401 root       1  59    0  128M   30M sleep 175:34  0.00% Xsun
 1418 root       3  58    0 4904K 4256K sleep   3:43  0.00% rpc.nisd
 1502 root       1  58    0 2120K 1248K sleep   3:19  0.00% atis
  193 root       6  53    0 3464K 1776K sleep   3:00  0.00% syslogd
  395 root       1  58    0 2816K 1576K sleep   1:16  0.00% xdm
 1416 root       3  58    0 1952K 1544K sleep   0:54  0.00% nis_cachemgr
 1500 root       1  59    0 2152K 1264K sleep   0:39  0.00% aarpd
  340 root       1  59    0 2240K 1592K sleep   0:21  0.00% _upsd
  202 root       1  48    0 1872K 1488K sleep   0:17  0.00% cron
 1410 root       6  18    0 2040K 1368K sleep   0:13  0.00% keyserv
```

“SIZE” refers to the total memory size of the job, while “RES” refers to the amount of memory actually in current use. The top process runs in the foreground (the display updates every 5-10 seconds); one needs to enter a 'q' or control-c to exit top.

If you want to terminate a job while it is running, you can either press control-C, or if it is running in the background

```
>kill %[job number]
```

where 'job number' is an integer found by typing the command 'jobs'.

The Fastcap manual does a good job of describing how to prepare input files, however the manual is 60 pages long, so I will provide a brief summary. First of all, something not covered in the manual, is how to edit a file in a UNIX environment. Most consider EMACS to be the best, most powerful text editor. There is a limited EMACS manual which can be accessed by typing

```
>man emacs
```

It is useful to note that all UNIX commands have a manual page, for example, 'man ps' lists, among other useful information, all the possible switches for the 'ps' command. A more extensive list of EMACS and UNIX commands is available in Daniel Gilly's book, UNIX in a Nutshell (O'Reilly & Associates, Inc. 1994). The two most important keystrokes for EMACS are control-x control-s (press control-x then control-s) which saves your work, and control-x control-c, which exits EMACS.

There are three different ways to specify objects that Fastcap can interpret. The authors recommend PATRAN, an AutoCAD style three-dimensional modeling program. I was not permitted the luxury of purchasing PATRAN primarily because of its $20,000 price tag. The second method is not as convenient as PATRAN, and that is entering the three-dimensional coordinates of conductors and dielectrics by hand. Unfortunately, this would require hours of manually entering sets of points. Fastcap is very fastidious about the order of the points and will hang if a single coordinate is entered out of order. Also, a simple object, due to edge panels and different discretization levels can very easily exceed a megabyte worth of data. Luckily, the third method of specifying objects is with a program called Cubegen that comes with Fastcap. Cubegen is a panel generating program that is capable of producing any rectangular object. The x, y and z heights are controlled with the -xh, -yh and -zh switches respectively. The discretization of the object, that is how small the panels are that make up each side of the object (more panels the more accurate), is specified with the -n switch (default is 3). (See page 3 of the Fastcap manual for examples of different discretizations.) for example, the command

```
>cubegen -nl -xhle-6 -yhlOOe-9 -zh5e-9 > xlmic.qui
```

produces a l um x lOO nm x 5 nm wire with its long axis along the x-axis. The '>' at the end of the line is a useful command which means dump the output of the previous operation into a file. In this case, the output of cubegen is written to the file xlmic.qui. I recommend naming the files something that easily identifies them. The name 'xlmic.qui' indicates a 1 micron long wire lying along the x-axis. The extension '.qui' has to be attached to the end of all object files; this identifies them as such for Fastcap.

To run Fastcap on the above file, xlmic.qui, the line

```
>fastcap xlmic.qui &
```

is entered (recall that the & at the end of the line runs the program in the background). If you want to use FFTCap, replace 'fastcap' in the above line with 'fftcap'. A typical Fastcap (FFTCap interface looks slightly different as will be discussed below) output for the above file looks like

```
Running fastcap 2.0 (18Sep92)
   Input: xlmic.qui
   Input surfaces:
   GROUP1

xlmic.qui, conductor title: 'le-06mXle-07mX5e-09m cube (n=1
  e=0.1)' outer permittivity: 1 number of panels: 460 number
  of extra evaluation points: 0 translation: (0 0 0) Date:
  Thu Aug 8 18:24:24 1996 Host: fenris.physics.haverford.edu

INPUT SUMMARY
  Expansion order: 2
  Number of partitioning levels: 5
  Overall permittivity factor: 1
  Total number of panels: 460
    Number of conductor panels: 460
    Number of dielectric interface panels: 0
    Number of thin conductor on dielectric interface panel:0
```

```
  Number of conductors: 1
Percentage of multiplies done by multipole: 83.1%

ITERATION DATA

Starting on column 1 (l%GROUPl)
1 2 3 4 5 6

CAPACITANCE MATRIX, attofarads
                    1
l%GROUP1 1       16.51
```

When there have been no errors, most of the information is not important. The capacitance matrix is explained in detail on page 22 of the Fastcap manual. Another important piece of information is the total number of panels. This is an indication of how much memory is needed to complete the calculation. A rule of thumb is that 4500 panels = 100 Mb of RAM. If the total number of panels is greater than the amount of available RAM, the computer will use virtual RAM (a tremendous amount of work for the hard drive, and VERY slow). The FFTCap output is similar to the above Fastcap output, except there is a great deal more information one has to sort through to find the two important figures. For this reason, I use Fastcap on very simple geometries that require very little computational power. On the other hand, Fastcap does a very poor job on more complex geometries, so I always use FFTCap when ten or more pieces are involved. Also, when I think I may have made a mistake specifying the orientation of the pieces (explained in the next paragraph), I use FFTCap because it has slightly better error-checking, as opposed to Fastcap which generally crashes.

Thus far I have explained how to produce objects with Cubegen. The next step in making more complex geometries involves list files. A list file provides a means of entering geometries with multiple conductors and dielectrics and a way to build more complicated geometries out of simpler structures. The following is an example of a list file

```
>more smallfry26.1st
G lOnm and lOOnm junctions in a square configuration
*
*
* left piece
C aylOOmics.qui 1 O O O +
C bylOOmics.qui 10 O O O +
C aylOOmics.qui 1 O le-6 O +
C bylOOmics.qui 10 O le-6 O +
C alOOmics.qui 1 .le-6 O O +
C blOOmics.qui 10 .le-6 O O
*
* right piece
C alOOmics.qui 1 .2e-6 1.7e-6 O +
C blOOmics.qui 10 .2e-6 1.7e-6 O +
C alOOmics.qui 1 1.2e-6 1.7e-6 O +
C blOOmics.qui 10 1.2e-6 1.7e-6 O +
C aylOOmics.qui 1 2.2e-6 le-6 O +
C bylOOmics.qui 10 2.2e-6 le-6 O +
C aylOOmics.qui 1 2.2e-6 O O +
C bylOOmics.qui 10 2.2e-6 O O +
```

```
C a9mics.qui 1 2.11e-6 0 0 +
C b9mics.qui 10 2.11e-6 0 0 +
C alOOmics.qui 1 l.lle-6 0 0 +
C blOOmics.qui 10 1.11e-6 0 0
```

All list files must start with a 'G' followed by a title. Any line the begins with * is a comment and is ignored by Fastcap. The general format of a list file is straightforward. There are two different ways to specify an object: as a conductor-dielectric interface or a dielectric-dielectric interface, indicated by a 'C' end 'D' respectively. For conductors, the first piece of information after the 'C' is the file name of the object, followed by the dielectric constant of the surrounding dielectric, followed by translation coordinates. The translation coordinates allow the user to translate objects wherever they want them. After some of the translation coordinates it should be noted that there is a '+'. This indicates that the object specified in the next line is considered part of the same object. The above list file has two large conductors made up of 6 and 12 other objects. The line format of a dielectric-dielectric interface is slightly different for which I refer the reader to page 8 of the Fastcap manual. Running Fastcap on a list file is done with a '-l' switch, i.e.

```
>fastcap -lsmallfry26.1st &
```

It is very easy to make a mistake when writing list files. I recommend using FFTCap because of the improved error-checking. The two most common errors involve mistakes with '+' signs and with translation coordinates. If you have made a mistake and used '+' to indicate that two different objects are part of the same entity, but are not in physical contact, FFTCap will report a non-convergence error. I recommend carefully reviewing the list file, and if something is not obviously wrong, check the object files to make sure that the objects specified are actually of the assumed dimensions (i.e. perhaps something is shaped differently than your thought it was). The other common error is problems with translation coordinates: two objects occupying the same space. FFTCap will report

```
placeq : Warning, removing identical quadrilateral panel
```

errors. Again, the best thing to do is review the list file, then the object files. If FFTCap reports any other errors, page 38 of the Fastcap manual has a list of warnings and errors with explanations.