KLAYOUT capacitors extraction with FASTERCAP : an algorithm

The parasitic capacitors extraction needs to start from the extracted netlist to get :

- the shape of each node : to build the 3D view of each node
- the node of each node : to integrate the extracted capacitors in the extracted netlist with the proper node names and connection

Create a "3d-shapes.txt" file

Get all layers : nwell, active, poly, contact, Metal1, (Viax, Metalx)*number of metal layers GDS number and heights positions, passivation height position

For **each_net** of the extracted netlist :
    Write into "3d-shapes.txt" file : "* shapes of the net_name"
    For each **layer** of **shapes_of_net** :
        If layer is not nwell nor active : ## *contact , vias and metals*
            Write into "3d-shapes.txt" file : "* shapes of the net_name in layer"
            For each shape of that layer :
                Zb = bottom of layer
                Zh = top of the layer
                If **layer** is not contact or Viax :
                    Split the **shapes** of shape_of_net of this **layer** into quadrilater Q : ## *decompose_trapezoids ?*
                    For each Q : ## *all coordinates are in meter*
                        Write into "3d-shapes.txt" file : "Q **net_name** (4 truples X,Y,Z of coordinates of Q)"
                        Write into "3d-shapes.txt" file : "Q **net_name** (4 truples of coordinates of Q)"
                    End_for # *each Q*
                End_if
                For each **point** of the **layer shapes** of , except the last one :
                    X2, Y2 = coordinates of the next points of the **shape**
                    Write into "3d-shapes.txt" file : "Q **net_name** X1 Y1 Zh X1 Y2 Zh X1 Y1 Zb X1 Y2 Zb"
                End_for ## *each point*
            End_for ## *each shape*
        Else : ## *layer is nwell or active*
            Z = 0
            For each shape of that layer :
                Split the **shapes** of shape_of_net of this **layer** into quadrilater Q : ## *decompose_trapezoids ?*
                For each Q : ## *all coordinates are in meter*
                    Write into "3d-shapes.txt" file : "Q **net_name** (4 truples of coordinates of Q)"
                    Write into "3d-shapes.txt" file : "Q **net_name** (4 truples of coordinates of Q)"
                End_for ## *each Q*
            End_for ## *each shape*
        End_if
        substrate = cell area – nwell – active ## *Z is still 0 and the net_name is 0*
            Split the **passivation** into quadrilater Q : ## *decompose_trapezoids ?*
            For each Q : ## *all coordinates are in meter*
                Write into "3d-shapes.txt" file : "Q **0** (4 truples of coordinates of Q)"
                Write into "3d-shapes.txt" file : "Q **0** (4 truples of coordinates of Q)"
            End_for ## *each Q*
    End_for ## *each layer*
End_for ## *each net*

Create a "3d-passivation.txt" file

Write into "3d-passivation.txt" file : "* passivation of **cell**"
Zb = bottom of topMetal + passivation thickness
Zh = top of the topMetal + passivation thickness
Top_passivation = topMetal sized by the thickness of passivation
Bottom_passivation = call area – Top_passivation
For each shapes of Top_passivation :
    Write into "3d-passivation.txt" file : "* shapes of the net_name in layer"
    Split the **shapes** of Top_passivation into quadrilater Q : *## decompose_trapezoids ?*
    For each Q : *## all coordinates are in meter*
        Write into "3d-shapes.txt" file : "Q **net_name** (4 truples of coordinates of Q)"
        Write into "3d-shapes.txt" file : "Q **net_name** (4 truples of coordinates of Q)"
    End_for
    For each **point** of the **layer shapes** of , except the last one :
        X2, Y2 = coordinates of the next points of the **shape**
        Write into "3d-shapes.txt" file : "Q **net_name** X1 Y1 Zh X1 Y2 Zh X1 Y1 Zb X1 Y2 Zb"
    End_for
End_for
For each shapes of Bottom_passivation :
    Write into "3d-passivation.txt" file : "* shapes of the net_name in layer"
    Split the **shapes** of Bottom_passivation into quadrilater Q : *## decompose_trapezoids ?*
    For each Q : *## all coordinates are in meter*
        Write into "3d-shapes.txt" file : "Q **net_name** (4 truples of coordinates of Q)"
        Write into "3d-shapes.txt" file : "Q **net_name** (4 truples of coordinates of Q)"
End_for
Write into "3d-passivation.txt" file : "Q passivation Xmin Ymin Zpass Xmin Ymax Zpass Xmax Ymax Zpass Xmax Ymin Zpass"

Create a "FasterCap_input.lst" file

Write into "FasterCap_input.lst" file : "0 Title : Faster shapes of **cell**"
Write into "FasterCap_input.lst" file : "* conductive shapes of the cell :"
Write into "FasterCap_input.lst" file : "C 3d-shapes.txt 4.2 0.0 0.0 0.0"
4.2 = dielectric constant of SiO2 = outer dielectric of the Metals = could be a parameter
Write into "FasterCap_input.lst" file : "* passivation of the cell :"
Write into "FasterCap_input.lst" file : "D 3d-passivation.txt 1.0 4.2 0.0 0.0 0.0"

Simulate with FasterCap the "FasterCap_input.lst" file : `FasterCap -b FasterCap_input.lst`

Parse the result to get the "Maxwell Capacitance Matrix", convert it to spice netlist format, include the capacitor netlist to the extracted netlist.

ANNEXES

FasterCap Input Files

Input File Syntax

This chapter provides reference information about the FasterCap input file syntax. FasterCap supports 3D or 2D capacitance extraction. The input file syntax is as similar as possible for the two cases.

3D Syntax

The input files specify the geometry of the conductors and of the dielectrics through a description of their surfaces, modeled as quadrilateral or triangular panels.

The input file syntax for FasterCap is 100% compatible with the FastCap2 "generic file format". It also introduces some enhancements, mainly the support of complex permittivity values and of hierarchical models.

Input File Structure

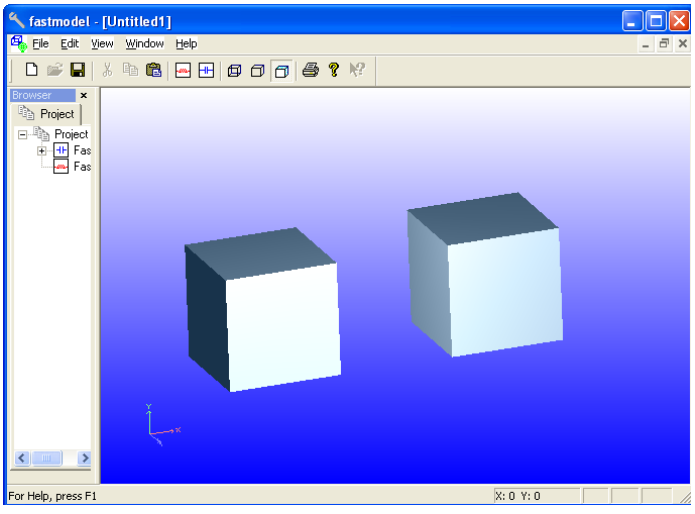
The input file is a sequential list of materials definitions (conductor 'C' statements and dielectric 'D' statements) and of geometric panel descriptions (triangular patch 'T' and quadrilateral patch 'Q').
Moreover, comments are allowed ('*' elements) as well as naming definitions ('N' elements)

Definitions can be nested in a hierarchical fashion to be able to re-use the geometric descriptions and to ease the construction of the models.
Panel geometric definitions ('Q' and 'T' elements) in the root file are assumed by default to belong to conductors embedded in a dielectric medium with unit permittivity.

All input files begins with a comment line. For compatibility and to ease possible file merging, you are recommended to start the line with a '*' character; anyway this is not mandatory, and the first line of any input file is simply ignored.

Remark: since FasterCap is able to automatically refine the geometry, it is best working with **an input mesh as coarse as possible**. This is a main difference with FastCap2, that needs a carefully pre-refined input mesh in order to produce accurate results.

An input file example is shown here below.



File *cubes.txt*:

```
* Array of two cubes, in air
*
* conductor | dielectric | offset
* file name | constant   | in space
*
C cube.txt    1.000000     0.0 0.0 0.0
C cube.txt    1.000000     2.0 0.0 0.0
```

The above input file references hierarchically the file *cube.txt*. This file contains a geometric description of a unit cube, as shown here below.

File *cube.txt*:

```
* 1mX1mX1m unit cube
*
* conductor     |   3D coordinates of the four corners
* name to which |   of the quadrilateral Q patch
* the Q patch   |
* belongs       |
*
Q mycube          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 mycube          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 mycube          1.0 0.0 0.0  0.0 0.0 0.0  0.0 0.0 1.0  1.0 0.0 1.0
Q mycube          0.0 0.0 0.0  0.0 1.0 0.0  0.0 1.0 1.0  0.0 0.0 1.0
Q mycube          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 mycube          0.0 0.0 1.0  1.0 0.0 1.0  1.0 1.0 1.0  0.0 1.0 1.0
```

The following sections explain in detail the meaning of each statement.

Conductor definitions ('C' statement)

```
Syntax: C <file> <outperm> <xoffset> <yoffset> <zoffset> [+]
```

The 'C' element at the beginning of a line defines a conductor. The geometry of the conductor surface is further specified in the file `<file>`. The conductor is assumed embedded in an homogeneous dielectric medium with relative permettivity `<outperm>`. The relative permittivity `<outperm>` can be complex valued, in the format *ere*-j*eim*, where *ere* is the real part of the complex permittivity value, and *eim* is the imaginary part. The conductor can be translated with respect to the coordinates defined in `<file>` by an offset *(xoffset, yoffset, zoffset)*, thus allowing to reuse the same geometric definitions multiple times.
The optional `+` argument is used to merge the current conductor with the next one, thus treating them a single conductor. With this option, panels with the same conductor name in different files will belong to the same conductor (see panels definitions for more information about conducor naming).

Example 1:

```
C cube.txt    1.000000     2.0 0.0 0.0
```

This input file fragments specifies a conductor whose geometry is defined in the file `cube.txt`, embedded in a dielectric with permettivity equal to one (e.g. air), and translated to the position *(2.0, 0.0, 0.0)*.

Example 2:

```
C cube.txt    1.000000     0.0 0.0 0.0 +
C cube.txt    1.000000     2.0 0.0 0.0
```

This input file fragments specifies a conductor made of two cubes in free air. It is considered a single conductor since the first statement ends by +, thus collating the two elements; i.e. the two cubes are considered short-circuited together.

Example 3:

```
C cube.txt    3.0-j0.02     2.0 0.0 0.0
```

This input file fragments specifies a conductor whose geometry is defined in the file `cube.txt`, embedded in a dielectric with complex permettivity equal to 3.0-j0.02 (e.g. with real part equal to 3.0 and imaginary part equal to 0.02), and translated to the position *(2.0, 0.0, 0.0)*.

Dielectric definitions ('D' statement)

```
Syntax: D <file> <outperm> <inperm> <xoffset> <yoffset> <zoffset> <xref> <yref>
<zref> [-]
```

The 'D' element at the beginning of a line defines a dielectric. The geometry of the dielectric surface is further specified in the file `<file>`. The dielectric surface is intended as the interface between two regions with relative permettivities `<outperm>` and `<inperm>`. The relative permittivities `<outperm>` and `<inperm>` can be complex valued, in the format *eRe*-j*eIm*, where *eRe* is the real part of the complex permittivity value, and *eIm* is the imaginary part.
The dielectric can be translated with respect to the coordinates defined in `<file>` by an offset *(xoffset, yoffset, zoffset)*, thus allowing to reuse the same geometric definitions multiple times.
The reference point *(xref, yref, zref)* and the optional - argument are used to specify which side of the dielectric interface has which permittivity. More specifically, the reference point is assumed to lie on the `<outperm>` side of all the panels in `<file>`. The optional - argument indicates that *(xref, yref, zref)* instead lies on the `<inperm>` side. The reference point is not translated, i.e. the offset only applies to the elements specified in `<file>`.

Remark: it is the user's responsibility to make sure that the reference point is on the same side of the dielectric interface for **all** panels. That is, each panel is evaluated as stand-alone with respect to the reference point, to define its `<outperm>` and `<inperm>` sides. There is no concept of a external and internal side of a surface specified by a group of panels, even if specified in the same file with the same conductor names, since FasterCap maintains no topological information.

For complex shaped surfaces, FasterCap supports also the option to specify the reference point panel by panel, see 3D Triangular panel definitions ('T' statement) and 3D Quadrilateral panel definitions ('Q' statement) for details about the syntax. You can also mix the two type of panel definitions, i.e. panels without and panels with a per-panel reference point specification. In this case, panels without a reference point specified per panel will use the reference point specified for the whole dielectric interface in the 'D' statement.

Example 1:

```
D sphere.txt  1.0 2.0  0.0 0.0 0.0  0.0 0.0 0.0  -
```

This input file fragments specifies a dielectric interface whose geometry is defined in the file *sphere.txt*. The interior of the sphere is filled with a material with relative permittivity equal

2.0, since the reference point is centered on the origin and the optional - argument is specified; while the dielectric medium ouside the sphere has a relative permittivity equal to 1.0.

Example 2:

```
D sphere.txt  1.0 3.0-j0.02  0.0 0.0 0.0  0.0 0.0 0.0  -
```

This input file fragments specifies a dielectric interface whose geometry is defined in the file *sphere.txt*. The interface is between air (relative permittivity equal to 1.0) and a lossy dielectric with complex relative permettivity equal to 3.0-j0.02 (e.g. with real part equal to 3.0 and imaginary part equal to 0.02).

Quadrilateral panel definitions ('Q' statement)

```
Syntax: Q <condname> <x1> <y1> <z1>   <x2> <y2> <z2>   <x3> <y3> <z3>   <x4> <y4>
<z4>   [<xref> <yref> <zref>]
```

The 'Q' element at the beginning of a line defines a quadrilateral panel. If the quadrilateral panel belongs to a conductor surface, the `<condname>` parameter is used to associate the panel to a parent conductor named `<condname>`. If the quadrilateral panel belongs to a dielectric surface, the `<condname>` parameter is ignored. The *(x1, y1, z1)*, *(x2, y2, z2)*, *(x3, y3, z3)*, *(x4, y4, z4)* 3D points specify the coordinates of the four corners of the panel; the coordinates must be entered in clockwise or counterclockwise order starting from any one corner.
In case the panel is part of a dielectric interface, you can use the optional coordinates *(xref, yref, zref)* to specify a reference point; see 3D Dielectric definitions ('D' statement) for more details about dielectric interfaces.

Example 1:

```
Q mycube  1.0 1.0 0.0  1.0 0.0 0.0  1.0 0.0 1.0  1.0 1.0 1.0
```

This input file fragments specifies a face of the surface of a conducting cube, whose name is `mycube`.

Example 2:

```
Q onecube  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 othercube  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 onecube  1.0 0.0 0.0  0.0 0.0 0.0  0.0 0.0 1.0  1.0 0.0 1.0
```

This input file fragments specifies two face of the surface of a conducting cube, whose name is `onecube`, and one face of the surface of a second conducting cube, whose name is `othercube`.

Example 3:

```
Q mycube  1.0 1.0 0.0  1.0 0.0 0.0  1.0 0.0 1.0  1.0 1.0 1.0  0.0 0.0 2.0
```

This input file fragments specifies a face of the surface of a dielectric cube, whose name is `mycube`, with a panel-specific dielectric reference point at (0,0,2).

Triangular panel definitions ('T' statement)

`Syntax: T <condname> <x1> <y1> <z1>  <x2> <y2> <z2>  <x3> <y3> <z3>  [<xref> <yref> <zref>]`

The 'T' element at the beginning of a line defines a triangular panel. If the triangular panel belongs to a conductor surface, the `<condname>` parameter is used to associate the panel to a parent conductor named `<condname>`. If the triangular panel belongs to a dielectric surface, the `<condname>` parameter is ignored. The *(x1, y1, z1)*, *(x2, y2, z2)*, *(x3, y3, z3)* 3D points specify the coordinates of the three corners of the panel; the coordinates must be entered in clockwise or counterclockwise order starting from any one corner.
In case the panel is part of a dielectric interface, you can use the optional coordinates *(xref, yref, zref)* to specify a reference point; see 3D Dielectric definitions ('D' statement) for more details about dielectric interfaces.

Example 1:

```
T mypyramid  1.0 1.0 0.0  1.0 0.0 0.0  1.0 0.0 1.0
```

This input file fragments specifies a face of the surface of a conducting pyramid, whose name is `mypyramid`.

Example 2:

```
T onepyramid  0.0 0.0 0.0  1.0 0.0 0.0  0.0 1.0 0.0
T otherpyramid  0.0 1.0 0.0  1.0 1.0 0.0  1.0 1.0 1.0
T onepyramid  0.0 0.0 0.0  1.0 0.0 0.0  0.5 0.5 1.0
```

This input file fragments specifies two face of the surface of a conducting pyramid, whose name is `onepyramid`, and one face of the surface of a second conducting pyramid, whose name is `otherpyramid`.

Example 3:

```
T mypyramid  1.0 1.0 0.0  1.0 0.0 0.0  1.0 0.0 1.0  0.0 0.0 2.0
```

This input file fragments specifies a face of the surface of a dielectric pyramid, whose name is `mypyramid`, with a panel-specific dielectric reference point at (0,0,2).

Name definitions ('N' statement)

`Syntax: N <oldcondname> <newcondname>`

The 'N' element at the beginning of a line renames a conductor. More specifically, it associates all the panels previously assigned to the conductor named `<oldcondname>` to the conductor named `<newcondname>`.

Example:

```
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  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  1.0 0.0 0.0  0.0 0.0 0.0  0.0 0.0 1.0  1.0 0.0 1.0
Q 1  0.0 0.0 0.0  0.0 1.0 0.0  0.0 1.0 1.0  0.0 0.0 1.0
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 1.0  1.0 0.0 1.0  1.0 1.0 1.0  0.0 1.0 1.0
N 1  mycube
```

This input file fragments specifies a cubic conductor whose name is '`1`'. The conductor is then renamed to a more user-friendly name `mycube`.

Single Input File option

FasterCap supports also the option to specify all input data in a single file. This avoids the need for a possibly large number of different input files.

The format is as follows:

```
(main input file)
End
File <first_sub-file>
(contents of <first_sub-file>)
End
File <second_sub-file>
(contents of <second_sub-file>)
End
```

In this way, all references to external files can be replaced by internal references; for instance, if you write

```
C first_sub-file 1.0 0.0 0.0 0.0
```

FasterCap will not try to access the file 'first_sub-file' from the file system, but will take the data from the 'File' section as per the above definition.

FasterCap allows hierarchical definitions, i.e. 'first_sub-file' can contain references to 'second_sub-file'. If the referenced files are not found internally, FasterCap assumes that they reside on the file system, and will look for them accordingly.

The characters following 'E' in 'End' and following 'F' in 'File' are optional. Like the other statements, 'E' and 'F' are case-insensitive, and must be the first character of the line in which they are defined.
Moreover, the 'End' statement in itself is optional, and is included only for input file readibility.

You can find some examples of single input files in the 'Samples' directory, for both 3D and 2D models.

Running FasterCap

Launching FasterCap from the shell

FasterCap can be launched in GUI-less mode from a System shell (e.g. a DOS shell under MS Windows, a KSH under Linux, etc.). To enter the GUI-less text mode, you need to pass the additional command option '-b' ('batch' mode) when launching FasterCap from the System shell.

For example:

```
FasterCap -b cube.txt -a0.01
```

Please note that the command options can be passed in arbitrary order. Please note also that the default option for the GUI-less mode is not Automatic, as it is instead in the GUI mode; so you need to specify the -a option with the relevant tolerance to trigger the Automatic solution mode.

For a list of valid running command options, please see the explanation in the *Run Menu dialog box* paragraph. The command options are indicated together with the relevant settings descriptions. Example: **'Stop when relative error is lower than (-a)'** indicates that the relevant parameter to control this setting when launched from a System shell is '-a'

You can also get a list of the valid command options with a short description when launching FasterCap from a shell with the '-?' parameter (or with any invalid unsupported parameter).

For example:

```
FasterCap -b -?

Usage: fastercap <input file> [-a<relative error>] [-ap]
                 [-m<mesh>] [-mc<mesh curvature] [-t<tolerance>]
                 [-d<interaction coeff>] [-f<outofcore>] [-g]
                 [-pj] [-ps<dimension>] [-o] [-r] [-c] [-i] [-v]
DEFAULT VALUES:
  -a: Automatically calculate settings, stop when relative error is lower than = 0.01

  -ap: Automatic preconditioner usage
  -m: Mesh relative refinement value = 0.1
  -mc: Mesh curvature coefficient = 3
  -t: GMRES iteration tolerance = 0.01
  -d: Direct potential interaction coefficient to mesh refinement ratio = 1
  -f: Out-Of-Core free memory to link memory condition = 5
  -g: Use Galerkin scheme
  -pj: Use Jacobi Preconditioner
  -ps: Use two-levels preconditioner with dimension = 5
OPTIONS:
  -o: Output refined geometry in FastCap2 format
  -r: Dump Gmres residual at each iteration
  -c: Dump charge densities in output file
  -i: Dump detailed time and memory information
  -v: Verbose output
  -b: Launch as console/shell application without GUI
  -b?: Print console usage (this text)
  -bv: Print only the version
  -h: Open only the help system (Linux only)
```

The list of valid return (exit) codes is:

```
// no error, normal end
#define FC_NORMAL_END 0
// catchall for generic error
#define FC_GENERIC_ERROR 1
// command line usage error
#define FC_COMMAND_LINE_ERROR 64
// cannot open input file(s)
#define FC_CANNOT_OPEN_FILE 66
// out of memory
#define FC_OUT_OF_MEMORY 71
// generic file error
#define FC_FILE_ERROR 74
// license error (e.g. invalid license)
#define FC_LICENSE_ERROR 77
// cannot go out-of-core (probably disk full)
#define FC_CANNOT_GO_OOC 97
// unknown exception caught
#define FC_EXCEPTION_ERROR 98
// user-requested break (not valid for console mode)
#define FC_USER_BREAK 125
```

The return codes can be checked under Linux with the following command:

```
echo $?
```

and under MS Windows with the command:

```
echo %ERRORLEVEL%
```

FasterCap Output

Output Iteration information

The iteration section provides the information related to the current solve iteration, in case of Automatic settings. In case of Manual settings, there will be only one section, for a single iteration corresponding to the chosen settings.

```
Iteration number #4 ***************************

**************************************
Increasing the geometric refinement..
Refinement completed
Mesh refinement (-m): 0.0131762
**************************************
Computing the links..
Number of panels after refinement: 480
Number of links to be computed: 45824
Done computing links
**************************************
Precond Type(s) (-p): Jacobi
GMRES Iteration: 0 1 2 3 4
GMRES Iteration: 0 1 2 3 4
Capacitance matrix is:
Dimension 2 x 2
g1_mycube 8.26757e-011 -2.73819e-011
g2_mycube -2.73807e-011 8.26804e-011

Weighted Frobenius norm of the difference between capacitance (auto
option): 0.00994789

Solve statistics:
Number of input panels: 12 of which 12 conductors and 0 dielectric
Number of input panels to solver engine: 24
Number of panels after refinement: 480
Number of potential estimates: 22912
Number of links: 46304 (uncompressed 230400, compression ratio is 79.9%)
Max recursion level: 18
Max Mesh relative refinement value: 0.0131762
Iteration time: 0.092000s (0 days, 0 hours, 0 mins, 0 s)
Iteration allocated memory: 12530 kilobytes
```

In the following, each subsection of the iteration section will be examined

Prologue subsection

```
Iteration number #4 ***************************
**************************************
Increasing the geometric refinement..
Refinement completed
```

```
Mesh refinement (-m): 0.0131762
************************************
Computing the links..
Number of panels after refinement: 480
Number of links to be computed: 45824
Done computing links
************************************
Precond Type(s) (-p): Jacobi
```

The prologue contains:

- The Mesh refinement value as automatically calculated by FasterCap
- The total number of panels and links after refinement
- The selected preconditioner type

Solve subsection

```
GMRES Iteration: 0 1 2 3 4
GMRES Iteration: 0 1 2 3 4
```

The solve subsection shows the status of the current calculation. The iterations will go on until convergence of the iterative linear system solution is achieved, within the required relative accuracy as set with the -t parameter (see 'Gmres tolerance' in the above Prologue subsecton and the Run Menu dialog box paragraph).

There will be one row of iterations for each separate conductor described in the input file.

If the -r global option is selected (see the Run Menu dialog box paragraph), also the residual at each iteration will be provided, as in the example below:

```
GMRES Iteration: 250 0 3.33 1 0.2 2 0.0145 3 0.00653 4 0.002
GMRES Iteration: 250 0 3.33 1 0.2 2 0.0146 3 0.00656 4 0.002
```

Result subsection

Capacitance matrix is:
Dimension 2 x 2
g1_mycube 8.26757e-011 -2.73819e-011
g2_mycube -2.73807e-011 8.26804e-011

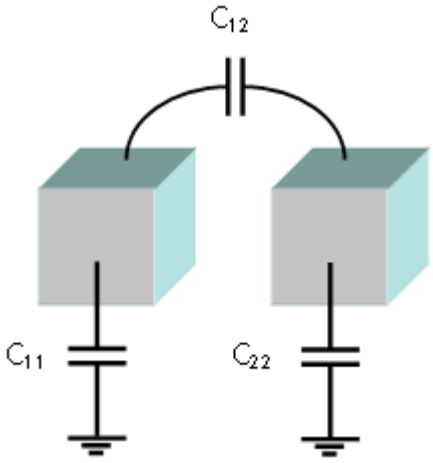
The result subsection contains the actual matrix capacitance resulting from the current iteration.

Besides the information about the matrix dimension, each row of the capacitance matrix will report the information about the conductor name. The conductors referred in the columns follow the same order of the rows (the matrix is, anyway, symmetric).

The format of the output matrix is that of a Maxwell capacitance matrix.

- 3D case

This means that with respect to the follwing figure, showing the actual self and mutual capacitances between the two cubes considered in the example:



the resulting matrix has the form:

$$\begin{bmatrix} C_{11}+C_{12} & -C_{12} \\ -C_{21} & C_{22}+C_{21} \end{bmatrix}$$

It follows that:

C11 = 8.268e-011 -2.738e-011 = 5.530e-011 = 55.30 pF
C12 = C21 = -(-2.738e-011) = 2.738e-011 = 27.38 pF

- 2D case

In the 2D case, the potential at infinity is not zero (the potential decreases as log(1/r), where 'r' is the distance from the source). Therefore, the potential is always referenced to one of the conductors, which is assumed to have zero potential (ground conductor). This last conductor is therefore not explicitly included in the Maxwell capacitance matrix row / columns.

Remember also that the unit of measurement of the capacitance matrix in this case is F/m.

Statistics subsection

```
Weighted Frobenius norm of the difference between capacitance (auto
option): 0.00994789

Solve statistics:
Number of input panels: 12 of which 12 conductors and 0 dielectric
Number of input panels to solver engine: 24
Number of panels after refinement: 480
Number of potential estimates: 22912
Number of links: 46304 (uncompressed 230400, compression ratio is 79.9%)
Max recursion level: 18
Max Mesh relative refinement value: 0.0131762
Iteration time: 0.092000s (0 days, 0 hours, 0 mins, 0 s)
Iteration allocated memory: 12530 kilobytes
```

The statistics subsection contains information about data used in the current iteration. In particular:

- The weighted Frobenius norm. The weighted Frobenius norm is used to measure the difference of the capacitance matrix resulting from the current iteration with respect to the capacitance matrix resulting from the previous iteration. This is the standard way to calculate the relative difference between two matrices. For instance, a value of 0.00994789 means a difference of 0.99% between the current result and the result in the previous iteration.
- The total number of input panels, as found in the input files structure
- The total number of input panels to the solver engine. This can be different from the total number of input panels, when a pre-processing of the input geometry is needed. The most common case is to transform the quadrilateral panels into two triangular panels, since the internal engine only deals with triangular elements. Other pre-processings deal with degenerate panels removal or thin panels regularization.
- The total number of panels, after mesh refinement. These are the actual panels over which the solution is calculated.
- The total number of potential estimates. This is the number of potential calculations used to build up the potential matrix, whose inversion will provide the capacitance matrix.
- The total number of links. This is the number of compressed matrix elements in the potential matrix. Calling 'N' the total number of panels (after mesh refinement), a full matrix would have a dimension equal to N x N. Thanks to compression, the actual matrix dimension is much smaller. The compression ratio is shown for reference between parenthesis.
- The maximum recursion level. This is the maximum recursion level reached in the meshing process when building the binary tree of panels.
- The maximum Mesh relative refinement value found during discretization (could be lower than the Mesh relative refinement value set by the User, in case the input geometry is already fine enough to be below the threshold set by the user. In this case, to force a further discretization, you need to specify a value smaller than the maximum Mesh relative refinement value found during discretization)
- The time elapsed and the memory allocated for this iteration.

outstanding performance best

The Maxwell Capacitance Matrix

White Paper

WP110301

FastFieldSolvers
Revision 03
November 2023

Copyright 2020 E. Di Lorenzo, FastFieldSolvers
All Rights Reserved
https://www.fastfieldsolvers.com

The Maxwell Capacitance Matrix

E. Di Lorenzo, FastFieldSolvers

***Abstract* – The meaning of the Maxwell capacitance matrix is reviewed, highlighting the rationale of this matrix form in relation to the internal mechanisms of a capacitance field solver. Some practical examples also show how to use the Maxwell capacitance matrix to build a SPICE netlist.**

I. INTRODUCTION

The capacitance matrix calculated by FasterCap, FastCap2 and by most capacitance field solvers is in the form of a Maxwell capacitance matrix.
The Maxwell capacitance matrix topic is well explained in any good electromagnetic text book, but since this is a widely asked question, this paper will present the theory, with a practical approach.

II. RELATIONSHIP

A Maxwell capacitance matrix provides the relation between voltages on a set of conductors to charges on the conductors. For example, for a generic conductor set, the following relation is valid:

$$\boldsymbol{Q} = \boldsymbol{C}\cdot\boldsymbol{V} \tag{1}$$

where C is the Maxwell capacitance matrix, and V and Q are voltage and charge vectors respectively. The relation between the elements of C (Maxwell capacitance matrix) and the physical capacitances between a set of conductors will now be derived.
Consider for example a four conductors case, as shown in Fig. 1. Here the mutual and auto capacitances (capacitances towards infinity) are explicitly drawn.

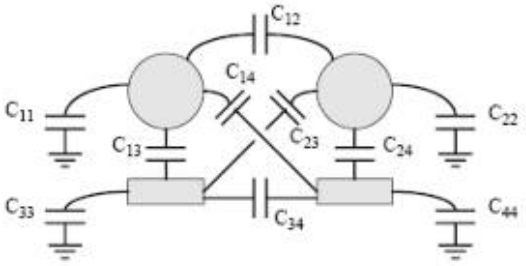


Fig. 1. Capacitances between four conductors

Physics tells us that the charge Q_1 on the conductor #1, given the voltages V_1, V_2, V_3, V_4 on the four conductors, is:

$$\begin{aligned} Q_1 = {} & C_{11}\cdot V_1 + C_{12}\cdot(V_1 - V_2) + \\ & + C_{13}\cdot(V_1 - V_3) + C_{14}\cdot(V_1 - V_4) \end{aligned} \tag{2}$$

which can be arranged as

$$\begin{aligned} Q_1 = {} & (C_{11} + C_{12} + C_{13} + C_{14})\cdot V_1 + \\ & - C_{12}\cdot V_2 - C_{13}\cdot V_3 - C_{14}\cdot V_4 \end{aligned} \tag{3}$$

So the first row of the Maxwell capacitance matrix **C** is

$$\left| C_{11}+C_{12}+C_{13}+C_{14} \;\; -C_{12} \;\; -C_{13} \;\; -C_{14} \right| \tag{4}$$

Completing the exercise, the full matrix has the form:

$$\begin{vmatrix} C_{11}+C_{12}+ \\ +C_{13}+C_{14} & -C_{12} & -C_{13} & -C_{14} \\ -C_{21} & C_{21}+C_{22}+ \\ & +C_{23}+C_{24} & -C_{23} & -C_{24} \\ -C_{31} & -C_{32} & C_{31}+C_{32}+ \\ & & +C_{33}+C_{34} & -C_{34} \\ -C_{41} & -C_{42} & -C_{43} & C_{41}+C_{42}+ \\ & & & +C_{43}+C_{44} \end{vmatrix} \tag{5}$$

Extending to the general case, the Maxwell capacitance matrix has the form:

$$\begin{vmatrix} C_{11}+C_{12}+ \\ +\cdots+C_{1n} & -C_{12} & \cdots & -C_{1n} \\ -C_{21} & C_{21}+C_{22}+ \\ & +\cdots+C_{2n} & \cdots & -C_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ -C_{n1} & -C_{n2} & \cdots & C_{n1}+C_{n2}+ \\ & & & +\cdots+C_{nn} \end{vmatrix} \tag{6}$$

It is worth noting that the capacitance values to be used on a SPICE-like simulator are the C_{xy} values, and not directly the values appearing as elements

Copyright 2020 E. Di Lorenzo, FastFieldSolvers
All Rights Reserved

in the Maxwell capacitance matrix. For instance, in our four conductor example, the element in the first column, first row of the Maxwell capacitance matrix is $C_{11}+C_{12}+C_{13}+C_{14}$, while in SPICE you should define a capacitor between node 1 and GND with the value C_{11}, a capacitor between node 1 and node 2 with the value C_{12}, etc. This means that some simple operations are in general needed to derive the C_{xy} values from the Maxwell capacitance matrix.

III. MOTIVATION

The reason why the capacitance field solvers, like FastCap2 and FasterCap, calculate the capacitance matrix in the form of a Maxwell capacitance matrix it that, since in general the relation (1) holds, to calculate the k-th column of **C**, you can set the potential of the k-th conductor to one (i.e. the k-th element of **V** is set to 1) and the potential of all remaining conductors to zero (i.e. all n-1 remaining elements of **V** are set to 0). The resulting **Q'** vector will be therefore equal to the k-th column of **C**.
In general, knowledge of the **C** matrix gives you all the elements needed to calculate the Q-V relation (i.e. the capacitance) for any set of voltages **V**.

IV. EXAMPLES

For instance, let's consider a Maxwell capacitance matrix for a two-conductors problem:

$$\begin{bmatrix} 4\,nF & -1\,nF \\ -1\,nF & 3\,nF \end{bmatrix} \tag{7}$$

From this matrix, we can easily find that:

$$\begin{aligned} C_{11} &= 4\,nF - 1\,nF = 3\,nF \\ C_{22} &= 3\,nF - 1\,nF = 2\,nF \\ C_{12} &= C_{21} = -(-1\,nF) = 1\,nF \end{aligned} \tag{8}$$

The corresponding SPICE netlist will then resemble the following text:

```
C11   node1 0       3n
C22   node2 0       2n
C12   node1 node2   1n
```

and is represented in Fig. 2.

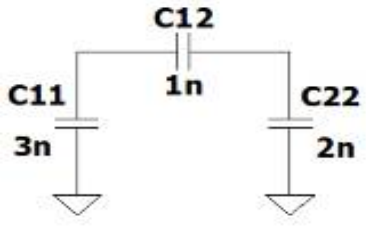


Fig. 2. Spice equivalent circuit

□

Copyright 2020 E. Di Lorenzo, FastFieldSolvers
All Rights Reserved