SPACE APPLICATION NOTE ABOUT GROUNDED 3D SUBSTRATE IMPLEMENTATION

S. de Graaf

Circuits and Systems Group
Faculty of Electrical Engineering
Delft University of Technology
The Netherlands

Report ET-CAS 02-01
February 21, 2002

Last revision: December 10, 2003.

1. INTRODUCTION

At Friday, 18 Jan 2002, i started the work for an implementation of a ‘‘grounded’’ 3D substrate resistance calculation method for the *space3d* program.

This calculation method can be useful for IC process technologies which have a ‘‘grounded’’ substrate. Currently, the ‘‘grounded’’ substrate can only be simulated by giving the second substrate layer a very high conductivity. Because *space3d* allows only the use of two substrate layers, this reduce the use of substrate layers to only one (first) substrate layer.

The 3D capacitance calculation method is also using a ground plane, and is using a calculation method based on Green’s functions, which maybe also can be used for the calculation of substrate resistance. Thus, we try to find out, if the implementation of ‘‘grounded’’ substrate can be based on the 3D capacitance calculation method. Tryouts by Eelco Schrik had shown that this method possibly also give correct substrate resistance results.

Note that the substrate can consist of different conduction layers (materials), which have each another conductivity value (sigma, Siemens/m). Thus, for capacitance calculation the layers above the substrate can have also different dielectric structure (epsilon, Farad/m). See also the "Space 3D Capacitance/Substrate Resistance Extraction" User’s Manuals for more details about the method used.

At Friday, 1 Feb 2002, there was a first working implementation and was i starting with this document.

1.1 Working Environment

A new SPACE tree was setup in directory:

```
/u/25/25/space2/Shadow/hp700.test/src/space
```

The source files have symbolic links to directory:

```
/u/25/25/space2/Shadow/general/src/space
```

A symbolic link is removed to ".old", when the source file is changed.
To compile the *Xspace (space3d)* program, do:

```
% cd /u/25/25/space2/Shadow/hp700.test
% make space6
```

The compilation result (executable) can be found in directory:

```
/u/25/25/space2/Shadow/hp700.test/src/space/space
```

Tests are done in project directory:

```
/u/25/25/space2/Shadow/hp700.test/src/space/sub3term/projectname
```

2. SUBSTRATE RESISTANCE EXTRACTION

Q: How are the substrate resistances calculated by *space3d*?
A: There are two methods:

1. The boundary-element method (provides accurate results, cost a lot of memory space and calculation time). Use option **-B** (*space3d* only). Note: Also called "3D substrate resistance extraction".

2. The interpolation method (results are not as accurate as the BEM method, requires less memory and is much faster as BEM) (to find the interpolation parameters, BEM is used for some small terminal configurations or the measurements results of standard configurations are used). Use option **-b** (can also be used with *space*).
   Note: Also called "inter. substrate resistance extraction".
   Note: It uses a Delaunay triangulation (2D mesh).

Note: BE method uses a (3D) BE mesh. Note: Substrate Resistance Extraction is always performed in flat extraction mode.
Set parameter "allow_hierarchical_subres" to do it explicit in hierarchical mode.
Note: For 3D substrate resistance extraction, the *space3d* element definition file (technology file) must contain substrate terminals.

Q: What are substrate terminals?
A: That are conducting polygons on top of the substrate, nodes between which the substrate resistances are connected. These terminals must be orthogonal (may not contain 45 degree edges/tiles).

Q: Which technology file elements can define substrate terminals?
A: Contacts, capacitances and transistors.

Note that adjacent substrate terminals are merged (creating a orthogonal polygon). This happens not always for BEM with parameter "sep_sub_term".

With @sub the area of the contact is the substrate terminal. With %(cond.list) the area specified by the mask cond.list is the terminal area. By transistors the bulk connection defines the substrate terminal.

```
sublayers:
#   name      conductivity        top_of_sublayer
#   [arb.id]  [real, Siemens/m]   [ <= 0 micron]
    first_lay    2.2              0.0  # top first  = top of substrate
    secnd_lay    2.8             -2.0  # top second = bottom first
```

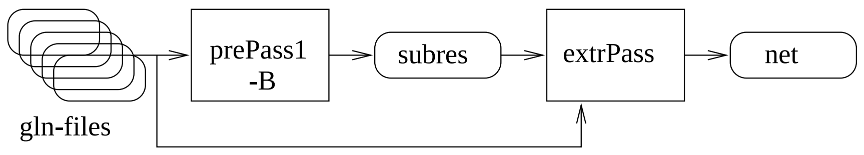
Note that maximum 2 different substrate layers can be specified. The top of the first substrate layer must start at 0.0 micron. The bottom of the last substrate layer is at -infinity. **Note that the sublayer top positions are always specified in micron**.

Note that it is not possible to specify a conducting backside or a finite substrate thickness. The thickness of the substrate is considered infinite. Default the substrate has also infinite dimensions in x- and y-direction. Optional the substrate edges (saw-lines) can taken into account.

Note that substrate parameters can be specified in the *space3d* parameter file. These parameters start with "sub3d." or are placed in a "BEGIN sub3d" block.

```
BEGIN sub3d
         saw_dist ...  # Edge effect parameters
         edge_dist ...
# Mesh parameters:
         max_be_area ...    ## no default!
         edge_be_ratio ...
END sub3d
```

The 3D substrate resistance calculations are done in the *space3d* prePass1. The results are used in the extrPass and possible combined with 3D capacitance results.



3. 3D CAPACITANCE EXTRACTION

Technology file specification requirements:

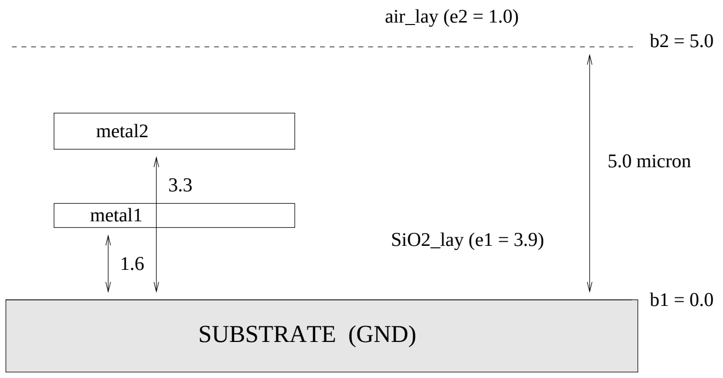
```
unit vdimension 1e-6

conductors :
#   name   : condition : mask : sheet-resistivity (ohm)
   metal1  :   cmf     :  cmf :  0    # ohm
   metal2  :   cms     :  cms :  0    # ohm

vdimensions:
#   name   : condition : mask : bottom thickness
   metal1v :   cmf     :  cmf :  1.6    1.0
   metal2v :   cms     :  cms :  3.3    1.2

dielectrics:
#   name      permittivity     bottom
#   [arb.id]  [real, epsilon]  [ >= 0 micron]
    SiO2_lay  3.9              0.0  # first  dielectricum
    air_lay   1.0              5.0  # second dielectricum
```

Note: The ground plane is present at the 0.0 position.
Note: The permittivity value is also called epsilon.



For 3D capacitance extraction (options **-c3** or **-C3**) there must be a technology file specification (see above) with ’vdimensions’ and ’dielectrics’ specification. The bottom of the first dielectric layer must start at 0.0 micron, what is the top position of the substrate ground plane. The top of the last dielectric layer is at infinity. **Note that the dielectric bottom positions are always specified in micron**. The material constants (permittivity) for the dielectric layers is always specified in F/m. At most three dielectric layers can be specified. No dielectric layers means ‘‘vacuum’’ (a special case).

4. 3D GROUNDED SUBSTRATE EXTRACTION

Technology file specification requirements:

```
sublayers:
#   name      conductivity     top_of_sublayer
#   [arb.id]  [real, Siemens/m]  [<= 0 micron]
    epi            10            0  # 1st sublayer
    epi2           40          -50  # 2nd sublayer
    metalization  1e5         -250  # 3th sublayer
```

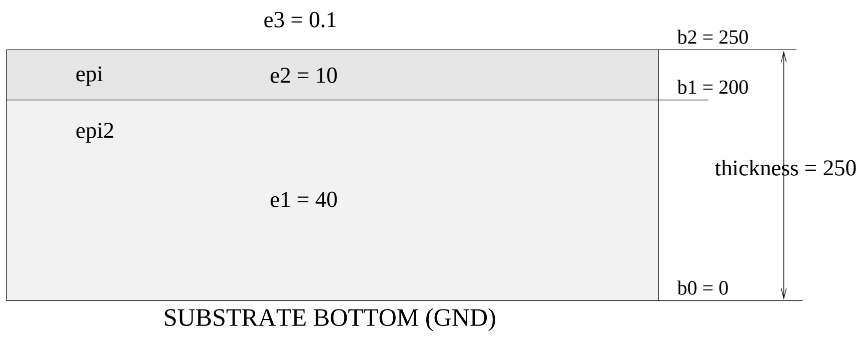
For grounded substrate extraction the last substrate layer (2nd or 3th sublayer) must be given the name "metalization". The conductivity value of this entry is not used. Default, a dielectric material constant is derived from the first sublayer by dividing it by 100 (in this case e3=0.1). Use the following specification in the *space3d* parameter file to get another value:

```
    sub3d.neumann_simulation_ratio   10000
```

In this case e3=0.001 is taken as material constant. Note that the top value of the "metalization" sublayer is the thickness of the substrate.
For grounded substrate extraction the capacitance 3D extraction method is used. Thus the "sublayers" technology information is rewritten to a "dielectrics" specification. To see this result, put in the *space3d* parameter file the following debug statement:

```
    debug.print_green_init
```

A figure of the rewritten above technology specification is given below:



5. NEW SPACE PARAMETERS

Two new parameters for a *space3d* parameter file are explained in the previous section. Here follows a list of all the new parameters.

5.1 debug.print_cap3d_init

Prints some initialization information of functions cap3dInitParam() and cap3dInit().
Note: A lot more can be printed, but is not implemented yet.

5.2 debug.print_green_init

Prints some initialization information of function greenInit().
Note: A lot more can be printed, but is not implemented yet.

5.3 debug.print_green_terms

Prints control flow and greenSeries debug information of function greenMpole(). Use this only for a very simple piece of layout (1 to 3 elements).

5.4 debug.print_green_gterms

Prints all green terms of each green term group of function greenMpole(). Use this only for a very simple piece of layout (1 to 3 elements).

5.5 min_divergence_term (default: 3)

The calculation of the green value stops normally when the contribution of the last term becomes less than "sub3d.green_eps" (default: 0.001). Note:

```
    if (Abs (term / value) < collocationEps) { ...; /* stop */ }
```

In some cases however, this green_eps is never reached. Sometimes this green_eps reach his smallest value after a number of image groups and then becomes bigger. This divergence is now build-in and the detection starts automatically after "min_divergence_term". Note that only this new parameter has no "cap3d." or "sub3d." prefix.

5.6 sub3d.max_green_terms

Not a new parameter, but also important to notice. This parameter is default 1 for nLayers < 2 and default 500 otherwise. This parameter can also be used as a (second) stop criterium. For nLayers=3 it is not wise to use more than 10 image groups, because each image group can have many terms.

5.7 sub3d.merge_images (default: on)

For nLayers=3 a new technique is implemented, which reduces the terms (images) in a image group. The terms are first sorted on distance value and when the distance for the same zq_sign are the same, then this terms (images) are merged. If you want, you can set this merge process "off" and experiment what happens.

5.8 sub3d.neumann_simulation_ratio (default: 100)

Only for grounded substrate extraction (see previous section).

5.9 sub3d.use_old_images (default: off)

For cap3d (greenType=0) and nLayers=2 a number of greenSeries() is improved (G21, G12 and G22). For sub3d (greenType=2) and nLayers=2 the greenSeries() is also improved. Also the directTerms are improved for greenType=2 and nLayers<=2. For sub3d (greenType=3) and nLayers=2 are also used the improved cap3d greenSeries(). For sub3d (greenType=3) and nLayers=3 are default the improved cap3d lowerMedium greenSeries() used (you can switch back to the not improved). You can use all old greenSeries() by putting the "use_old_images" parameter "on".

5.10 sub3d.use_lowest_medium (default: on)

For nLayers=3 and z values which are on the interface boundary between e2/e3 better the lowest medium greenSeries() can be used (because of convergence).

5.11 sub3d.use_mean_green_values (default: off)

Possibly better green result values can be calculated by putting this parameter "on". After the stop criterium, it tries to take the mean value of the last two groups term values.

6. BOUNDARY ELEMENT MODES

See the "Shape and Weight Functions" section in the "3D Cap. Extraction" and "Substrate Res. Extraction" manuals. The mode can be set with "be_mode" in a *space3d* parameter file. Default mode is "0c". Note that "collocation" may be used for "0c" and "galerkin" for "0g". The "be_mode" sets the following variables (see function greenInit() in file "green/init.c"):

be_mode	FeModeGalerkin	FeModePwl	pointGreenDist	collocationGreenDist
0c	FALSE	FALSE	infinity	0.0
1c	FALSE	TRUE	infinity	0.0
0g	TRUE	FALSE	infinity	infinity
1g	TRUE	TRUE	infinity	infinity

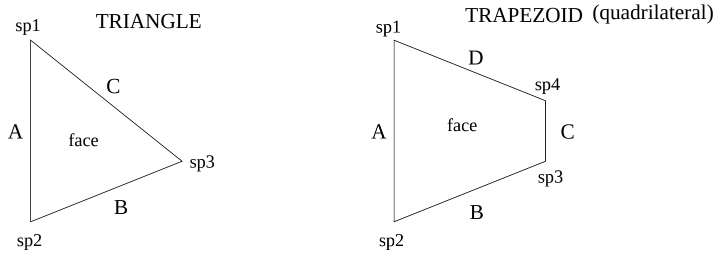
Note that the ratios pointGreenDist and collocationGreenDist are devided by microns for the greenCase SUBS. This is an inconsequent parameter behaviour, but it are undocumented parameters. The above *xxx*Dist values are the defaults. The values can be set with parameters "point_green" and "collocation_green".

Note that value asymCollocationGreenDist (default "infinity") can be set with parameter "asym_collocation_green". If the distance between two spiders is smaller than asymCollocationGreenDist, then function CollocationGreenPwc() is called twice!

See also the following section for a Green's function flow. But, in the default case, function green() calls greenPwc(). Function greenPwc() calls one of the functions PointGreen(), CollocationGreenPwc() or GalerkinGreenPwc() depending of the values distance 'd' and face length sum 'r':

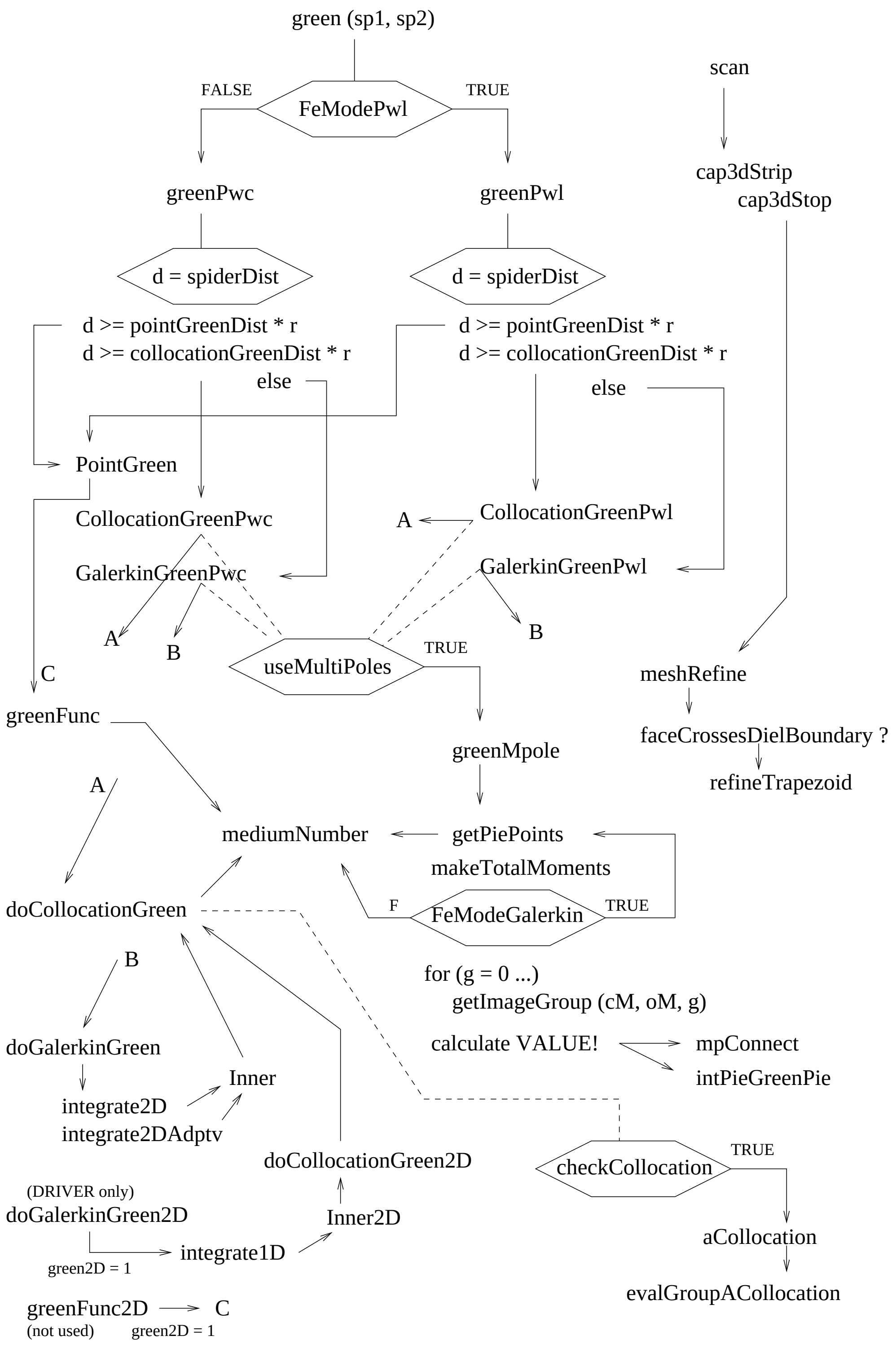
```
    r = sp1 -> face -> len + sp2 -> face -> len;
    d = spiderDist (sp1, sp2);
```

By function greenPwl() is 'r' the sum of the maximum spider distances (≥ 0). Note that both spiders must have a face in the above 'r' calculation. By ‘‘pwc’’, the spiders are center spiders (flagged by face ≠ 0). By ‘‘pwl’’, spiders are not center spiders (face ≡ 0). Function feSize() called by faceRecur() calculates the face length.



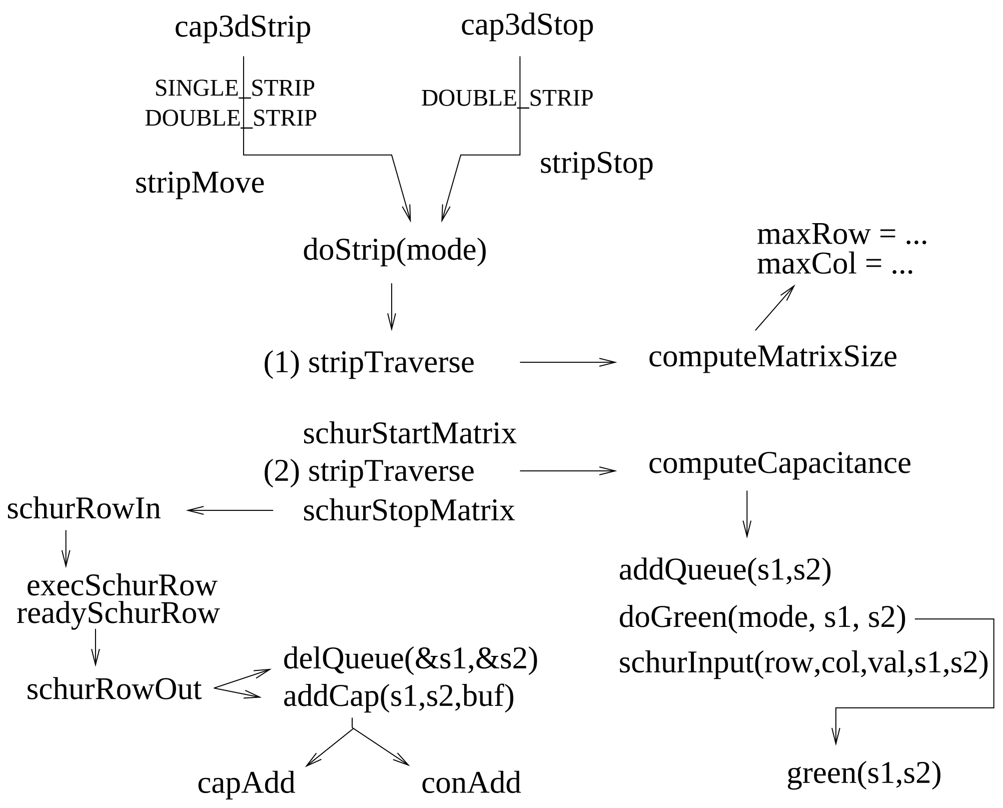
The face length is the maximum spider distance (≥ 0) around the face. Thus, by default *xxx*Dist parameters and be_mode "0c" the function CollocationGreenPwc() will be called by greenPwc(). Function CollocationGreenPwc() shall call greenMpole(), because parameter "use_multipoles" is default "on".

7. GREEN’s FUNCTIONS



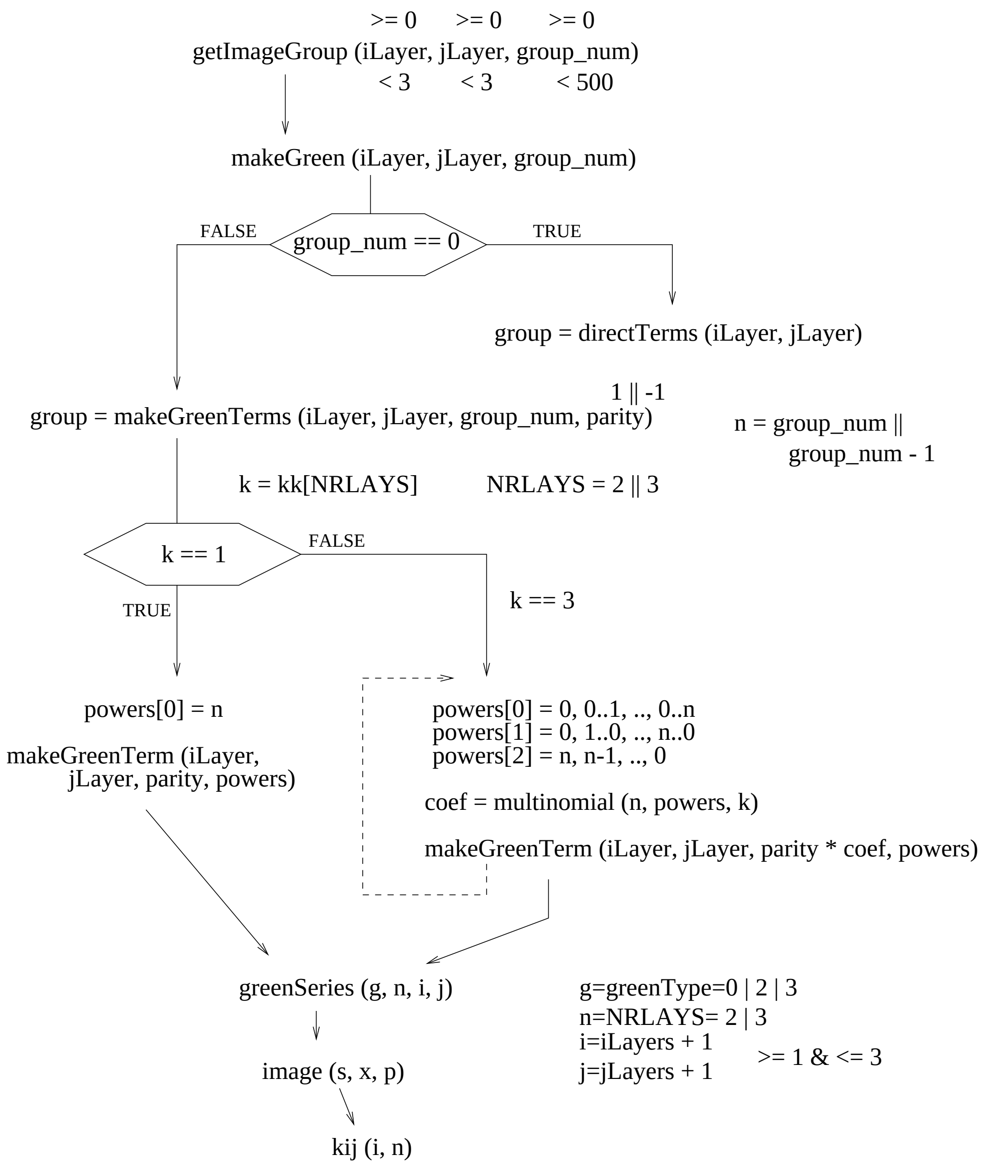
All the Green functions call function greenMpole() when variable useMultiPoles is TRUE (the default case). This is a fast numerical integration method build in by U. Geigenmuller. Function mpConnect() calculates default a value based on quadrupoles. This order can be set with parameter "mp_max_order" (default 2). Other choices are: monopoles (0), dipoles(1) or octopoles (3). I have only used the default value.

7.1 Function: doGreen()



By scanning function cap3dStrip() is called and this function calls function doStrip() twice for two strips (a single and double strip). Function doStrip() calculates a Schur matrix size and starts a strip traversal with function computeCapacitance(). Function doGreen() is called, which calls the green() function.
The green() function calculates a Green value which is put in the Schur matrix. The matrix row is inverted and the result is a capacitance or a conductance value.

7.2 Function: getImageGroup()


>= 0 >= 0 >= 0
getImageGroup (iLayer, jLayer, group_num)
< 3 < 3 < 500
makeGreen (iLayer, jLayer, group_num)
group_num == 0
FALSE
TRUE
group = directTerms (iLayer, jLayer)
1 || -1
group = makeGreenTerms (iLayer, jLayer, group_num, parity)
n = group_num || group_num - 1
k = kk[NRLAYS]
NRLAYS = 2 || 3
k == 1
FALSE
TRUE
k == 3
powers[0] = n
makeGreenTerm (iLayer, jLayer, parity, powers)
powers[0] = 0, 0..1, .., 0..n
powers[1] = 0, 1..0, .., n..0
powers[2] = n, n-1, .., 0
coef = multinomial (n, powers, k)
makeGreenTerm (iLayer, jLayer, parity * coef, powers)
greenSeries (g, n, i, j)
g=greenType=0 | 2 | 3
n=NRLAYS= 2 | 3
i=iLayers + 1
j=jLayers + 1
>= 1 & <= 3
image (s, x, p)
kij (i, n)

Function getImageGroup() returns an imageGroup_t data structure. This data structure contains a group 'size' and 'images' field. The 'images' field can be an array of image_t data structures. The 'size' field tells how many structures are in the array.
No array is returned (size ≡ 0), if

```
    1) nLayers[greenCase] < 2 and group_num >= 1
or  2) group_num >= GREEN_MAX_TERMS (500)
```

Thus, only for group_num ≡ 0 something is returned for nLayers[greenCase] < 2. Note that for group_num ≡ 0 always an array with one/two image_t structures (size ≡ 1 or 2) is returned. This situation is handled by function directTerms(). Note that for nLayers[greenCase] < 1 one image_t is returned. Note that maxGreenTerms is 1 by nLayers[greenCase] < 2. Note that iLayer and jLayer are the layer medium numbers.

Function getImageGroup() remembers each group in the imageHead array. Thus, a new group is only made by a call to function makeGreen() when the group size is zero. An image_t data structure (called a term) contains the following three fields:

```
    typedef struct image {
        green_t strength; real_t distance; int zq_sign;
    } image_t;
```

For group_num ≥ 1 (by nLayers[greenCase] ≥ 2) function makeGreenTerms() is called and that function calls makeGreenTerm() one or more times. For nLayers[greenCase] ≡ 2 only one greenSeries() is made. A greenSeries() consist out of 2 to 12 image_t structures (terms). For nLayers[greenCase] ≡ 3 function makeGreenTerm() is called one of more times depending of the group_num (n). For cap's (greenCase is DIEL) the value of n = group_num - 1 and also a swapping parity flag (1 or -1) is used in that case. The following powers and coef are used for n values 0 to 3:

```
         n = 0           n = 1           n = 2           n = 3
   powers[0][1][2] coef  [0][1][2] coef  [0][1][2] coef  [0][1][2] coef
          0  0  0   1     0  0  1   1     0  0  2   1     0  0  3   1
                          0  1  0   1     0  1  1   2     0  1  2   3
                          1  0  0   1     1  0  1   2     1  0  2   3
                                          0  2  0   1     0  2  1   3
       T(0) = 1                           1  1  0   2     1  1  1   6
n > 0: T(n) = n + 1 + T(n-1)              2  0  0   1     2  0  1   3
                                                          0  3  0   1
number of terms T(n) is:                                  1  2  0   3
  n : 0,1,2, 3, 4, 5, 6, 7, 8, 9,10,11,12, 13,..          2  1  0   3
T(n): 1,3,6,10,15,21,28,36,45,55,66,78,91,105,..          3  0  0   1
```

The coef value is calculated with function multinomial(). For the maximum powers value n the coef is always 1. Function multinomial() calculates a maximum coef value for each value of n, which after that is reduced by the powers[0][1][2] values. Only powers values > 1 gives maximum coef value reduction. And a powers value of n gives a maximum reduction (coef becomes 1). The other powers are in that case always 0 and give no reduction.

When the convergence criterium needs more groups (num_group) and more terms to calculate, the calculation time (and memory usage) grows rapidly. By DIEL, by num_group=11 (n=10) and a greenSeries() of 6 images the number of terms (group size) becomes 66 * 6 = 396.

Function kij(i,n) calculates (gives back) the n-th power for material constant a[i]:

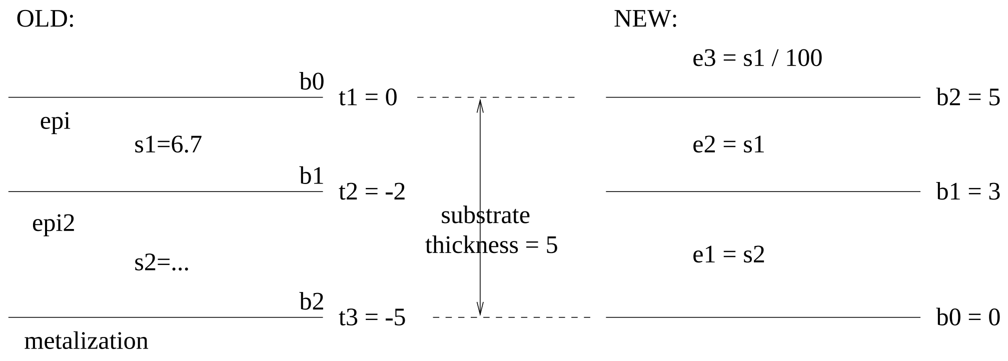
```
  a[i] = (matconst[i] - matconst[i+1]) / (matconst[i] + matconst[i+1]);
```

8. SPIDERS

For each conducting layer a mesh is generated. The highest layer must be the "@sub" (substrate) layer. And also the highest conductor layer (and number) must be that of the substrate. Like spiders in a web, *space3d* is using spiders as mesh points and in the center of a mesh face. Each spider has 3D coordinates (x,y,z). The distance between two spiders is calculated with function spiderDist(), as follows:

```
meshCoor_t spiderDist (sp1, sp2)
{
    x = sp2 -> act_x - sp1 -> act_x;
    y = sp2 -> act_y - sp1 -> act_y;
    z = sp2 -> act_z - sp1 -> act_z;
    return sqrt (x * x + y * y + z * z);
}
```

By substrate calculation (in prepass1) the spider z-coordinate is always zero, because the substrate contacts are laying on this surface. For the new calculation method all z-coordinates must be ≥ 0, with as reference the grounded substrate (metalization, z=0). Now, the substrate contact spider z-coordinates must be lifted with the substrate thickness. See figure:



By a first implementation i have added this thickness to the spider z-coordinates in function getPiePoints() when it was called by function greenMpole().
Now, i want add the thickness directly to the spiders, when they are created. Spiders are created in function enumPair() by a call to spiderPair(). See the figure:

```
enumPair (tile, newerTile, edgeOrien)
    spiderPair (tile, newerTile, edgeOrien, bdr)
        doSpiderPair (tile_l, tile_r, bdr, orientation)
            mesh = recogMesh (tile_l, tile_r, bdr, &i)
            for (; m = mesh[i]; i++)
                spiderFindNew (tile_l, tile_r, side, x, y, z, level, conductor)
```

Function spiderPair() is done in prepass1 for option **-B**, because flag optCap3D is TRUE. Function spiderFindNew() creates a new spider, when the spider can’t be found. The z-coordinates are set in function recogMesh(), but only when the tile has a substrate contact (solid_l|_r = TRUE).

9. EXAMPLE: single

By this grounded substrate experiment of Eelco Schrik the substrate resistance value is compared with the calculated capacitance value. Consist out of the following layout:

```
ms single
term cmf 125 154 278 313 a
box cmf 0 300 0 600
me
```

Because lambda is 0.01 micron, the first metal (cmf) box is 3 by 6 micron. Note that terminal ’a’ shall split the box in two pieces. Chosen technology file values are:

```
unit vdimension 1e-6

conductors:
  condIN   : cmf : cmf  : 0

contacts:
  cont_sub : cmf : cmf @sub : 0

vdimensions:
  metal1v  : cmf : cmf  : 250 0

dielectrics:
  SiO2  3.9      0
  air   0.001  250

sublayers:
  epi         10     0
  substrate  1e5  -250
```

Space parameter file "space.green.p" values:

```
sub3d.max_be_area    inf
sub3d.be_window      inf
cap3d.max_be_area    inf
cap3d.be_window      inf
```

Executing *space3d* for 3D substrate resistance extraction:

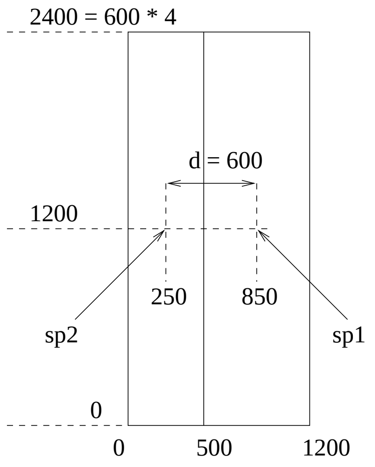
```
% space3d -B -P space.green.p -E space.green.t single
% xsls single
network single (terminal a)
{
    res 12.09232k (a, SUBSTR);
}
% cat layout/single/subres
c 1 1 xl 0 yb 0 g_sub 8.269711e-05
nr_neigh 0
```

Note that R_sub = 1 / g_sub.
Executing *space3d* for 3D capacitance extraction:

```
% space3d -C3 -P space.green.p -E space.green.t single
% xsls single
network single (terminal a)
{
    net {SUBSTR, a};
    cap 285.5633e-18 (SUBSTR, GND);
}
```

Note that g_sub / s1 must be equal to cap / (e1 * Epsilon0) =

```
285.5633e-18 / 3.9 * 8.855e-12 = 8.2736e-06
```



Three values are calculated in greenMpole:

value 1 for d = 0, r = 4800
spo = spc = sp1

value 2 for d = 600, r = 4800
spo = sp1, spc = sp2
spo = sp2, spc = sp1

value 3 for d = 0, r = 4800
spo = spc = sp2

9.1 -B calculation

For **-B** greenSeries(2,2,1,1) is done, because greenType=2, nlayers=2, iLayer=jLayer=0 (note: sp1.z=sp2.z=0). Contains the following two images:

```
       (str)     (distance) (zq_sign)
image(2*k(1,n)/s1, -2*b1*n, 1);
image(2*k(1,n)/s1,  2*b1*n, 1);

k(1,1) = (s1 - s2) / (s1 + s2) = -99990 / 100010 = -1
k(1,2) = k(1,1) ˆ 2 = +1
k(1,3) = k(1,1) ˆ 3 = -1
s1 = 10   b1 = -250*400 = -100000
strength = str * coef   (coef = 1)
```

n=group_num	strength	distance	zq_sign
0	+0.1	0	-1 / 1
1	ca. -0.2	+/-200000	1
2	ca. +0.2	+/-400000	1
3	ca. -0.2	+/-600000	1

Note that group number 0 is done by function directTerms() and generates also two images for nlayers=2 (strength=1/s1).

See function greenMpole() in file "green/mpgreen.c":

```
    convergenceRadius = cRadius + oRadius;
    R.x = oCenter.x - cCenter.x;
    R.y = oCenter.y - cCenter.y;

    value = 0;
    for (g = 0; g < maxGreenTerms; g++) {
        grp = getImageGroup (cMedium, oMedium, g);
        term = 0;
        for (i = 0; i < grp.size; i++) {
            Z = grp.images[i].zq_sign * cCenter.z + grp.images[i].distance;
            R.z = oCenter.z + Z;
            Rabs = R3Norm (R);
            distratio = Rabs / convergenceRadius;
            strength = grp.images[i].strength;
            if (distratio > multipolesMindist && distratio > 1) {
                gterm = strength * mpConnect (cMomMir, oMom, R, Rabs);
            }
            else { ...; /* use ‘traditional’ integration method */
                gterm = strength * (intPieGreenPie (...)).value;
            }
            term += gterm;
        }
        value += term;
        if (Abs(term/value) < collocationEps) return (TRUE);
    }
```

The above code is done for two spiders (spc, spo) which are laying in the center of a rectangle. It is also done for spc ≡ spo. In that case R.x = R.y = 0 and Rabs = Abs(R.z). Note that the ’c’ stands for ’charge’ and ’o’ for ’observation’ point. In our case cCenter.z = oCenter.z (= 0 for greenType=2).
Function mediumNumber() gets the cMedium(=iLayer) and oMedium(=jLayer) numbers, which are equal and in this case 0. The convergenceRadius = cRadius, because oRadius=0 for FeModeGalerkin ≡ FALSE. You can see that a green ’value’ is calculated in the for-loop for group numbers ’g’ 0 to maxGreenTerms - 1. For each group an image-group is get by function getImageGroup(). The value grp.size is the number of images. For each image the value ’gterm’ is calculated and for all images the value ’term’. Each group adds value ’term’ to the green ’value’.
There are two calculation methods for value ’gterm’. Depending on ’distratio’ a choice is made for the "multipole expansion" method or the "traditional integration" method. If the spider points are close together the "traditional" method is used. Note that the default value of ’multipolesMindist’ is 2.
For Center.z positions 0 is:

```
    R.z = Z = grp.images[i].distance = +/- 2 * b1 * n
```

For this example the ’zq_sign’ does not do anything. See appendix A for a listing of the result values of the *space3d* execution. Note that only some interesting ’gterm’ values are shown. The traditional integration method is done for group g=0 (value gt3). For all

other groups the multipole expansion method is done (value gt2). By this example function mpConnect() returns the same value for both images. This happens, because the Center.z positions are 0 and also the z positions of the multipole points are 0. Thus, the 3D substrate resistance method can be optimized for this situation.
The maximum number of groups (0 to 9 = 10) are used by green value (2a). Note that only one value (the mean value of 2a and 2b) is returned. This value is also listed and is:

```
    (0.000244566 + 0.000250645) / 2 = 0.000247605
```

It is maybe interesting what happens in the symmetric case, when the terminal splits the rectangle in the middle (at position x = 600). In that case, both values must be the same. Note that the split position is the terminal begin (left x) position.
For collocationEps=0.001 the following results can be found:

-B calculation: epi=10, substrate=1e5 at -250			
symmetry	resistance	groups	values (e-4)
400/800	12.08359k	6,10,9,5	4.65233,2.42537,2.54713,5.80035
500/700	12.09232k	6,10,9,5	4.87142,2.44566,2.50645,5.43058
600/600	12.09725k	5,10,10,5	5.12930,2.47163,2.47163,5.12930
1200	12.72528k	7	3.99777

-B calculation: epi=10, substrate=s2 at -250, symmetry 1200				
s2 value	resistance	groups	value(e-4)	res2 (k)
1e6	12.72528k	7	3.99777	12.71998
1e5	12.72528k	7	3.99777	12.71998
1e4	12.71480k	6	3.99447	12.7211
1e3	12.71591k	6	3.99482	12.72167
1e2	12.72214k	4	3.99678	12.72795

-B calculation: epi=10, substrate=1e5 at xp, symmetry 1200				
xp value	resistance	groups	value(e-4)	res2 (k)
-500	12.73802k	4	4.00177	12.74332
-300	12.72299k	6	3.99705	12.72829
-250	12.72528k	7	3.99777	12.71998
-200	12.70412k	8	3.99112	12.70979
-100	12.64837k	14	3.97360	12.65448
- 50	12.54996k	27	3.94269	12.54387
- 25	12.32976k	53	3.87351	12.3237

The 'res2' value is more precise (see the comment on the next page).

In the following results a grounded substrate is used. This is achieved by calling the second material "metalization". The green values are calculated with a two layer 3D capacitance calculation method.

-B calculation: epi=10, metalization(=0.1) at -250				
symmetry	res (k)	groups	values * e-9	res2 (k)
500/700	11.96955	14,19,18,14	482293,242355,247938,537260	11.97336
*500/700	11.97211	3,3,3,3	482474,242225,248024,537441	11.97466
1200	12.60033	15	395851	12.59416
*1200	12.59375	3	395644	12.59629

In the above table *space3d* has also calculated the resistance values for medium numbers iLayer=jLayer=0 (in place of 1). In that case we asume that the substrate contact is just laying in the "epi" layer. And the result is possible less depended of the second material constant. The result of this calculation seems to be good and (most important) there are less groups needed (total number of images is 30 and 10). See appendix B for a listing of both result values (for symmetry 1200) of the *space3d* execution. Note that the corrected resistance values (res2) are closer together.

Table with changing "neumann_simulation_ratio":

-B calculation: epi=10, metalization(=e2) at -250, symmetry 1200				
e2 value	resistance	groups	value	res2 (k)
1.0	11.56883k	5	0.000363445	11.56407
0.1	12.60033k	15	0.000395851	12.59416
0.01	12.71387k	49	0.000399418	12.70759
0.001	12.72544k	157	0.000399782	12.7191

The above resistance values are calculated with changing second material constant, which is depended of the following formule:

```
e2 = e1 / sub3d.neumann_simulation_ratio
```

The default "sub3d.neumann_simulation_ratio" is 100. In the above table, a higher ratio does not give a much better result. But a lot more groups are needed.

-B calculation: epi=10, metalization(=0.1) at xp, symmetry 1200						
xp value	res1 (k)	grps	value*e-9	term*e-9	res2 (k)	res3 (k)
-500	12.612	12	396206	334	12.61695	12.61632
-400	12.616	13	396341	344	12.61046	12.61105
-300	12.596	14	395726	382	12.60244	12.60183
-250	12.600	15	395851	388	12.59416	12.59473
-200	12.589	17	395496	356	12.58338	12.58383
-100	12.524	22	393447	373	12.52976	12.52939
- 50	12.426	29	390383	365	12.42046	12.42074

In the above table the metalization depth 'xp' is the changing factor. The resistance value 'res1' is standard calculated by *space3d*. As you can see, this value seems not to be correct, because the value at xp=-500 must be larger than at xp=-400. This happens, because the calculation is broken at the convergence criterium. And *space3d* is swapping around the ‘‘real’’ resistance (green) value. It make sens, at which group number (even or odd) the calculation is broken. I have tried to calculate a better resistance value with ’res2’ and ’res3’. The used formula are:

```
    value_res2 = value_res1 - term / 2
    value_res3 = value_res1 - term / 2 - (termold + term) / 4
```

Note that termold and term are different in sign (+/-). And Abs(termold) > Abs(term). Note that in general the calculation must never stop after the first group. Thus the value of collocationEps must always be < 1.0 and maxGreenTerms must always be ≥ 2.

9.2 -C3 calculation

For **-C** greenSeries(0,2,2,2) is done, because greenType=0, nlayers=2, iLayer=jLayer=1 (note: sp1.z=sp2.z=100000). This is the highest possible medium number, because the z position is not smaller than 100000 (the bottom of metal conductor starts in the highest medium). The greenSeries() contains the following two images:

```
        (str)       (distance)  (zq_sign)
image(-k(1,n+1)/e2, 2*t1*(n-1), 1)
image( k(1,n+1)/e2, 2*t1*(n+1), 1)
k(1,1) = (e1 - e2) / (e1 + e2) = 3.899 / 3.901 = 1
k(1,2) = k(1,1) ˆ 2 = +1
e1 = 3.9  e2 = 0.001  t1 = 250*400 = 1e5
coeff = coef * parity    (coef=1)
strength = str * coeff
```

group_num	n	coeff	strength	distance	Rabs	zq_sign
0	-	-	+/-1e3	0	0 /2e5	-1 / 1
1	0	1	ca. -/+1e3	-2e5/2e5	0 /4e5	1
2	1	-1	ca. +/-1e3	0/4e5	2e5/6e5	1
3	2	1	ca. -/+1e3	2e5/6e5	4e5/8e5	1
4	3	-1	ca. +/-1e3	4e5/8e5	6e5/1e6	1

Note that function mpConnect() divides by Rabs. Thus the smallest Rabs generates the biggest return value. And this is responsible for swapping (+/-) term values. See appendix C for a listing of the result values of the *space3d* execution.

-C3 calculation: SiO2=3.9 air=0.001 at 250			
symmetry	capacitance	groups	values (1e-4)
500/700	285.563e-18	89,124,123,85	12.4983,6.26892,6.42527,13.9221
600/600	285.451e-18	87,123,123,87	13.1494,6.34170,6.34170,13.1494
1200	271.691e-18	98	10.2391

-C3 calculation: SiO2=3.9 air=xx at 250 for symmetry 1200			
air value	capacitance	groups	value
0.001	271.6910e-18	98	0.001023913
0.01	272.0615e-18	31	0.001022519
0.1	278.5341e-18	10	0.000998757
0.2	285.3009e-18	7	0.000975069
0.5	306.1882e-18	5	0.000908552
1.0	341.0105e-18	4	0.000815776

Looking to the above results, you can see, that by C3 calculation a large number of groups is needed for convergency. This cost a lot of memory and execution time. Thus, the performance seems not to be so good. If we play with the value of the dielectric material constant for "air", you see that the number of groups becomes less. And a factor 100 does not change the capacitance value too much.

10. EXAMPLE: single with 3 material constants

In the "GREEN's FUNCTIONS" section we have seen that the greenSeries() used is depending on the number of material constant layers which are specified in the technology file.
Also are the greenSeries() for 3 layers more complicated. Because they contain more images. They are only known for the 3D capacitance extraction (see appendix D).
We want to use these greenSeries() for the grounded substrate resistance extraction case and have defined a new greenType=3 (see appendix E).
Another problem is, that function makeGreenTerm() expands this number of images a number of times depending on the group number (n) used. This is done for k=3 to get a power series (multinomial expansion).
The allocation of these images cost a lot of memory and is only possible till a certain group number. The default maxGreenTerms (= 500) can not be reached. And these many images cost also a lot of calculation time. Thus, fast convergence is an important issue for the nLayers=3 approach. Set maxGreenTerms for example to 10. This generates for the highest group number(=9) 55 * greenSeries images.

Now we run example C3 extractions with nLayers=3. The results are listed in the tables:

-C3 calculation: SiO2=3.9 SiX=2 at b1 air=0.1 at 250 for symmetry 1200					
b1 value	capacitance	groups	images	memory(Mb)	green_eps
10	146.1497e-18	8	506	0.189	0.025075
50	146.2258e-18	7	338	0.185	0.022848
125	146.3835e-18	7	338	0.185	0.021476
200	146.9480e-18	7	338	0.185	0.022507

-C3 calculation: SiO2=3.9 SiX=2 at b1 air=0.1 at 500 for symmetry 1200					
b1 value	capacitance	groups	images	memory(Mb)	green_eps
50	278.115e-18	2	12	0.177	0.000476
125	278.253e-18	2	12	0.177	0.000514
200	279.048e-18	3	42	0.177	0.000392
250	410.368e-18	3	42	0.177	0.000864
300	540.495e-18	3	42	0.177	0.000358
400	541.789e-18	3	42	0.177	0.000347

-C3 calculation: SiO2=3.9 SiX=2 at b1 air=0.001 at 500 for symmetry 1200					
b1 value	capacitance	groups	images	memory(Mb)	green_eps
50	278.103e-18	2	12	0.177	0.000561
125	278.245e-18	2	12	0.177	0.000456
200	279.036e-18	3	42	0.177	0.000418
250	410.351e-18	3	42	0.177	0.000938
300	540.466e-18	3	42	0.177	0.000406
400	541.762e-18	3	42	0.177	0.000402

Out of the above results, you can see, that there is no convergence problem when the last dielectricum starts at position 500 and the interconnect is laying at position 250. The change in the capacitance value is most sensible around this position. We can also try out what happens when we variate this interconnect position.

-C3 calculation: SiO2=3.9 SiX=2 at 250 air=0.001 at 500 for symmetry 1200					
int.pos.	capacitance	groups	images	memory(Mb)	green_eps
300	278.857e-18	3	42	0.177	0.000421
400	277.187e-18	3	42	0.177	0.000066
499	189.530e-18	3	42	0.177	0.000539
499.9	145.162e-18	3	42	0.177	0.000417
499.99	139.739e-18	3	42	0.177	0.000402
499.999	139.193e-18	3	42	0.177	0.000400
499.9999	139.138e-18	3	42	0.177	0.000400
500 alt.	139.132e-18	3	42	0.177	0.000400
500	xxxxxxxx	140	2743862	xxxx	xxxx

Thus, we see, that the calculation at and above position 500 divergent. But in practice there can not lay a part of the interconnect in the "air". When we don’t use greenSeries(0,3,3,3) for position 500, but as an alternative greenSeries(0,3,2,2) we can get a good(?) result.

Other trials:

-C3 calculation: SiO2=3.9 SiX=2 at 10 air=xx at 250 for symmetry 1200					
xx value	capacitance	groups	images	memory(Mb)	green_eps
1e-0	208.759e-18	5	122	0.177	0.000644
1e-1	146.150e-18	8	506	0.189	0.025075
1e-2	143.562e-18	6	212	0.181	0.337333
1e-3	xxxxxxxx	4	62	0.193	4.465231

In above results the calculation is stopped by the divergence criterium. By ’xx’ 1e-3 was the stop position incorrect. Maybe we must only stop after an even number of groups. See also appendix F for the results. In the next table you can see what happens when no divergence stop is build in:

-C3 calculation: SiO2=3.9 SiX=2 at 10 air=xx at 250 for symmetry 1200					
xx value	capacitance	groups	images	memory(Mb)	green_eps
1e-0	208.759e-18	5	122	0.177	0.000644
1e-1	134.586e-18	128	2097026	48.2	0.000911
1e-2	xxxxxxxx	140	2743862	xxxx	xxxx

For ’xx’ 1e-1 (after divergence) convergence is reached by 128 groups. For ’xx’ 1e-2 and 1e-3 the program runs out of memory after 140 groups.

10.1 Grounded substrate calculation

Let we now try some calculations for grounded substrate with 3 layers.

-B calculation: epi=10 epi2=20 at xp metalization at -250, symmetry 1200					
xp value	res2 (k)	groups	images	memory(Mb)	green_eps
-125	xxxxxxxx	140	2743862	xxxx	xxxx

Because of the above problems, we try the alternative for the substrate contact layer number:

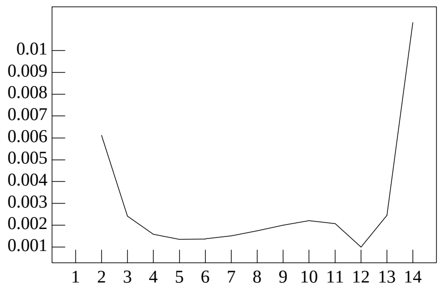
-B calculation: epi=10 epi2=xx at -125 metalization at -250, symmetry 1200					
xx value	res2 (k)	groups	images	memory(Mb)	green_eps
20	12.57990	3	42	0.185	0.000785
50	12.56698	3	42	0.185	0.000613
100	12.56225	3	42	0.185	0.000353
200	12.55987	3	42	0.185	0.000152
400	12.55869	3	42	0.185	0.000029

-B calc. with collocationEps=0.001: epi=10 epi2=40 at xp metalization at -250					
xp value	res2 (k)	groups	images	memory(Mb)	green_eps
-10	11.91610	3	42	0.185	0.000493
-125	12.56929	3	42	0.185	0.000692
-200	12.58779	5	202	0.189	0.000993
-210	12.59093	11	2202	0.236	0.000610
-211	12.59062	11	2202	0.236	0.000901
-212	12.58474	12	2862	0.248	0.000310
-213	12.58579	12	2862	0.248	0.000821
-214	12.59735	13	3642	0.267	0.000332
-215	12.59587	13	3642	0.267	0.000586
-215.3	12.59542	13	3642	0.267	0.000862
-215.4	12.59527	13	3642	0.267	0.000954
-215.43	12.59523	13	3642	0.267	0.000982
-215.44	12.59521	13	3642	0.267	0.000991
-215.448	12.5952	13	3642	0.267	0.000999
-215.4494	12.5952	13	3642	0.267	0.001000
-215.4495	xxxxx	118	2738192	xxxx	xxxx

-B calc. with divergence detection: epi=10 epi2=40 at xp metalization at -250					
xp value	res2 (k)	groups	images	memory(Mb)	green_eps
-210	12.58864	8	842	0.205	0.001105
-211	12.58877	8	842	0.205	0.001153
-212	12.58889	8	842	0.205	0.001202
-213	12.58902	8	842	0.205	0.001251
-214	12.58915	7	562	0.197	0.001299
-215	12.58932	7	562	0.197	0.001341
-215.4	12.58939	7	562	0.197	0.001358
-215.5	12.58941	7	562	0.197	0.001363
-216	12.58949	7	562	0.197	0.001384
-220	12.59016	7	562	0.197	0.001567
-230	12.59155	7	562	0.197	0.002147
-240	12.59267	8	842	0.205	0.003126
-249	12.59138	9	1202	0.213	0.007615
-249.5	12.59508	10	1652	0.220	0.009477
-249.9	12.59326	11	2202	0.236	0.015466

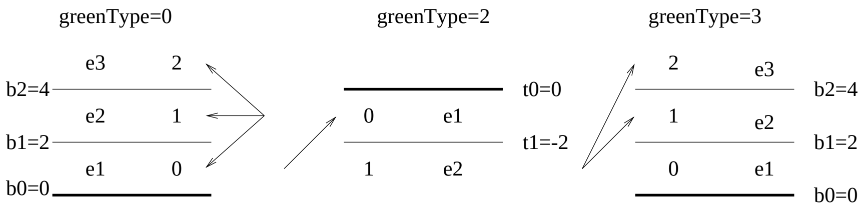
Out the above tables, we see that the stop criterium (collocationEps=0.001) not more is reached by 'xp' values $\geq$ -215.4495 (the program runs out of core). To overcome the above problem we can set another stop criterium. For example choice collocationEps > 0.001 or set the maxium number of groups to 13. But we can also build in a divergence detection and stop at that point. This is done in the last table.
The following figure shows the green_eps values for xp=-215.45 and appendix G lists the

result values of the program execution.



11. THE MEDIUM NUMBER

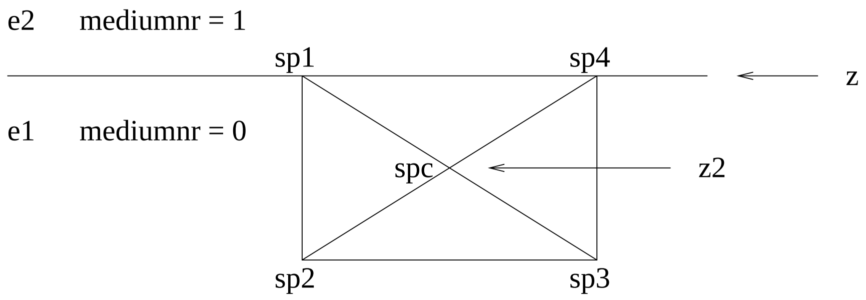
Function mediumNumber(z) returns the number of the dielectric medium in which <z> is located. For greenType=2 this is always 0 and for greenType=3 normally nLayers - 1, and for greenType=0 this is 0 to nLayers - 1. Note that for nLayers > 2 the function tries to use nLayers - 2 for z ≡ b2 (if parameter useLowestMedium="on" (default="on")), because the lower greenSeries gives better convergence results.



Function mediumNumber() validates also the <z> values. For substrate contacts all values must be on the subcontPosition. By capacitance extraction (greenType=0) the spiders in a face can have different <z> positions. This gives a problem (see the figure below), when sp1 or sp4 <z> is laying on the dielectric interface boundary. In that case the highest medium number is returned and not all spiders are laying in the same medium. The new function mediumNumber2(z,z2) solves this problem, see code:

```
int mediumNumber2 (z, z2) real_t z, z2;
{
    ...
    for (i = 1; i < nLayers[greenCase]; i++)
        if (z < matbounds[greenCase][i]) break;
    if (i > 1 && z == matbounds[greenCase][i-1] && z >= z2) --i;
    return (i-1);
}
```

This function returns the same medium number for <z> as given to <z2>. Note that for <z> ≡ <z2> the lowest medium number is returned. This is needed in function getPiePoints() to return TRUE in the following case:



The conductor does not really cross the dielectric interface, thus the multipole expansion method can be used. For spiders sp1 and sp4 the same medium number must be used as for spider spc.

12. UNDERSTANDING THE GREENSERIES

See thesis reports Z.Q. Ning (page 47-49) and N.P. van der Meijs (page 67-69). The greenSeries() of appendix D and E can be translated to the Green's function formula in the following way (see source files "mkgreen.c" and "images.h"):

```
#define greenSeries(gT, nL, i, j) if (greenType == gT \
   && nLayers[greenCase] == nL && iLayer == i-1 && jLayer == j-1)

#define image(s, x, p) (images[i].strength = coefficient * s, \
         images[i].distance = x, images[i].zq_sign = p, i++)

/* For g=1 to maxGreenTerms-1 */
greenSeries (3, 2, 1, 1) {
    image(-(k(1,1 + p(0))/e(1)), -2*t(1)*(1 + p(0)), -1);
    image(-(k(1,1 + p(0))/e(1)),  2*t(1)*(1 + p(0)), -1);
    image(  k(1,1 + p(0))/e(1) , -2*t(1)*(1 + p(0)),  1);
    image(  k(1,1 + p(0))/e(1) ,  2*t(1)*(1 + p(0)),  1);
}

/* For g=0, directTerms (iLayer, jLayer) */
    image(+(1/e(jLayer+1)), 0, -1);
    image(-(1/e(jLayer+1)), 0,  1); /* no '-' for greenType=2 */
```

To use the above greenSeries(): the greenType must be 3, the total number of dielectric layers must be 2 (greenCase=1) and both the source and observation points must be in medium numbers iLayer=0 and jLayer=0. In that case a number of image terms (4) is used. Note that p(0) is n = g-1 (0 to ..) for greenType ≠ 2 (or n=g for normal substrate calculation); g is the group number used in function greenMpole(); e(1)=e1 is the first dielectric constant value; t(1)=d is the end position of the first dielectricum (negative for greenType=2); k(1,p) calculates [(e(1)-e(2))/(e(1)+e(2))]ˆp or Kˆp.

Each image term has three arguments (s, x, p) which values are stored in an images[] array and can be reused for a given group number (g).

The coefficient = parity * coef (see makeGreenTerms() coef=k=1 for nLayers[]=2) See makeGreen(), if greenType ≠ 2 and Even(g), then parity=-1 else parity=1. Thus for nLayers[] is 2, the coefficient is -1 for Even(g). Thus for n=0 (g=1), the coefficient is 1 (also for the directTerms, g=0).
Now we can reconstruct the Green's function:

```
    G11(p,q) = 1 / (4 * PI * Epsilon0) * valueGreen
    valueGreen = { directTerms(g=0) + greenSeries(g=1 to ...) }
```

The Green's value is calculated for at most maxGreenTerms terms. But normally after some terms the calculation is stopped, because a new term contribution becomes insignificant small (stop criterium collocationEps).
Note that the zq_sign=-1 is used for the (zp - zq + distance) terms, and the zq_sign=+1 is used for the (zp + zq + distance) terms (see these Meijs, page 69).
Note that all terms must be divided by the same dielectric constant (in this case e1).

13. GREENSERIES IMAGE CONVERSION AND REDUCTION

The z-distance R.z between points p and q depends on the zq_sign used:

```
    zq_sign=-1 :   R.z = zp - zq + distance
    zq_sign=+1 :   R.z = zp + zq + distance
```

A greenSeries() image with zq_sign=-1 can be written (converted) to an image with zq_sign=+1.

```
    image (S, d1, -1) == image (S, d1 - 2*zq, 1)
or
    image (S,-d1, -1) == image (S,-d1 - 2*zq, 1)
or
    image (S, d1 + 2*zq, -1) == image (S, d1, 1)
or
    image (S,-d1 + 2*zq, -1) == image (S,-d1, 1)
```

Note that the sign of R.z makes no sense, thus:

```
    zq_sign=-1 :   R.z = -zp + zq - distance
    zq_sign=+1 :   R.z =  zp + zq + distance
    -zp + zq - d1 = zp + zq + d2
    d2 = - d1 - 2*zp

    image (S, d1, -1) == image (S,-d1 - 2*zp, 1)
or
    image (S,-d1, -1) == image (S, d1 - 2*zp, 1)
or
    image (S, d1 - 2*zp, -1) == image (S,-d1, 1)
or
    image (S,-d1 - 2*zp, -1) == image (S, d1, 1)
```

The only problem is, that this is only possible for a known value of zq or zp. By substrate calculation we know both zq and zp. For normal substrate calculation (greenType=2) we know that zq=zp=0. Thus:

```
  greenSeries (2, 2, 1, 1) {
    image(2*k(1,m)/s(1),-2*b(1)*m, 1);
    image(2*k(1,m)/s(1), 2*b(1)*m, 1);
  }
```

is equal to:

```
  greenSeries (2, 2, 1, 1) {
    image(4*k(1,m)/s(1),-2*b(1)*m, 1);
  }
```

For grounded substrate calculation (greenType=3) we know that zq=zp=t2. Thus:

```
    image(S,-2*t1*n - 2*t2*m, -1);
    image(S, 2*t1*n + 2*t2*m, -1);
```

is equal to:

```
    image(2*S, 2*t1*n + 2*t2*(m-1), 1);
```

Thus the images of greenSeries(0, 3, 2, 2):

```
  T1 = 2*t(1); u = p(0) + p(2);
  T2 = 2*t(2); m = p(1) + p(2); n = p(0) - p(2);

  image(-k(1,u+1)*k(2,m  )/e(2), T1*(n-1) + T2*(m),1);
  image( k(1,u+1)*k(2,m  )/e(2), T1*(n+1) + T2*(m),1);
  image(-k(1,u)  *k(2,m+1)/e(2),-T1*( n ) - T2*(m+1),-1); <= 2
  image(-k(1,u)  *k(2,m+1)/e(2), T1*( n ) + T2*(m+1),-1); <= 3
  image( k(1,u)  *k(2,m+1)/e(2),-T1*( n ) - T2*(m+1),1);
  image( k(1,u)  *k(2,m+1)/e(2), T1*( n ) + T2*(m+1),1);
  image(-k(1,u+1)*k(2,m+1)/e(2),-T1*(n-1) - T2*(m+1),-1); <= 6
  image(-k(1,u+1)*k(2,m+1)/e(2), T1*(n-1) + T2*(m+1),-1); <= 7
  image( k(1,u+1)*k(2,m+1)/e(2),-T1*(n+1) - T2*(m+1),1);
  image( k(1,u+1)*k(2,m+1)/e(2), T1*(n-1) + T2*(m+1),1);
```

can be rewritten for greenSeries(3, 3, 3, 3) to (taking together 2&3 and 6&7):

```
  image(-k(1,u+1)*k(2,m  )/e(2), T1*(n-1) + T2*(m),1); <= 0
  image( k(1,u+1)*k(2,m  )/e(2), T1*(n+1) + T2*(m),1);
  image(-2*k(1,u)*k(2,m+1)/e(2), T1*( n ) + T2*(m+1),-1);
  image( k(1,u)  *k(2,m+1)/e(2),-T1*( n ) - T2*(m+1),1);
  image( k(1,u)  *k(2,m+1)/e(2), T1*( n ) + T2*(m+1),1);
  image(-2*k(1,u+1)*k(2,m+1)/e(2), T1*(n-1) + T2*(m+1),-1); <= 5
  image( k(1,u+1)*k(2,m+1)/e(2),-T1*(n+1) - T2*(m+1),1);
  image( k(1,u+1)*k(2,m+1)/e(2), T1*(n-1) + T2*(m+1),1);
```

Also terms 0 and 5 above can be taken together.

Note that the value of the z-distance is more important for points with less lateral distance (see table). Thus convergence is better for short lateral distances.

R.x	R.z	sqrt(R.x^2 + R.z^2)	importance R.z	conv.factor
0	500	500	1.000	0.002
10	500	500.1	1.000	0.020
50	500	502.5	0.995	0.100
100	500	509.9	0.981	0.196
200	500	538.5	0.928	0.371
500	500	707.1	0.707	0.707
1000	500	1118	0.447	0.894
1200	500	1300	0.384	0.923
2000	500	2061.5	0.243	0.970
3000	500	3041.4	0.164	0.986
5000	500	5024.9	0.100	0.995

14. REFERENCES

- Accurate and Efficient Modeling of Global Circuit Behavior in VLSI Layouts, Z.Q. Ning, thesis 20-Apr-1989.
- Models for Large Integrated Circuits, P.M. Dewilde, Z.Q. Ning, Kluwer Academic Publishers.
- Reduced Models for the Behavior of VLSI Circuits, A.J. van Genderen, thesis 3-Oct-1991.
- Accurate and Efficient Layout Extraction, N.P. van der Meijs, thesis 27-Jan-1992.
- Green's Functions for Capacitance Calculations (Draft), N.P. van der Meijs, 25-Oct-1994.
- Hybrid Models for Parasitic Capacitances in Advanced VLSI Circuits, E.B. Nowacka, thesis 27-Jan-1997.
- Multipole Acceleration of Numerical Integration in the BE Method for 3D Capacitance Extraction, U. Geigenmuller, N.P. van der Meijs.
- Space 3D Capacitance Extraction User's Manual.
- Space Substrate Resistance User's Manual.
- Space Programming Notes, S. de Graaf, report 24-Jul-2001.

14.1 Source Files Changes

— substr/subcont.c (4.15, 5/2 AvG) - Prevent that Rsub may become < 0.

— spider/cap3d.c (4.34, SdeG) - Debug print statements.

— spider/recogm.c (4.14, SdeG) - Spider subcontZpos not always 0.

— green/init.c (4.19, SdeG) - Set greenType, z-bounds and subcontZpos.

— green/green.c (4.11, SdeG) - Debug print statements.

— green/mpgreen.c (4.9, SdeG) - Divergence stop, print statements, etc.

— green/mkgreen.c (4.15, SdeG) - Added greenType 3, mediumNumber2(), etc.

— green/images.h (4.2, SdeG) - Added images for greenType 3, changed images!

— green/images.c - Removed!

Note: Checked in Monday, Apr 8 16:17 2002.

15. APPENDICES

APPENDIX A -- Example: single, -B listing

```
cap3dInitParam: prePass1=1 optSubRes=1 greenCase=1 greenType=2
greenInit: reading substrate specification (fromParamFile=0)
greenInit: i=1 n=3 m='epi'       e=10     b=0
greenInit: i=2 n=3 m='substrate' e=100000 b=-250
greenInit: greenData.nlayers[greenCase]=2 maxGreenTerms=500 collocationEps=0.001
greenPwc(): d=0 r=4800 sp1=(850,1200,0) sp2=(850,1200,0) ------------------------------ (1)
greenMpole(): spo=(850,1200,0) spc=(850,1200,0) convergenceRadius=1250
g=0 i=0 str= 0.1000 dis=0       zq=-1 Rabs=0      distratio=0   gt3= 2.443e-4 t= 2.443e-4
g=0 i=1 str= 0.1000 dis=0       zq= 1 Rabs=0      distratio=0   gt3= 2.443e-4 t= 4.887e-4
greenMpole(0): value= 0.000488708 term= 0.000488708 term/val=1.000000
g=1 i=0 str=-0.1999 dis= 200000 zq= 1 Rabs=200000 distratio=160 gt2=-9.997e-7 t=-9.997e-7
g=1 i=1 str=-0.1999 dis=-200000 zq= 1 Rabs=200000 distratio=160 gt2=-9.997e-7 t=-1.999e-6
greenMpole(1): value= 0.000486709 term=-0.000002000 term/val=0.004108
g=2 i=0 str= 0.1999 dis= 400000 zq= 1 Rabs=400000 distratio=320 gt2= 4.997e-7 t= 4.997e-7
greenMpole(2): value= 0.000487708 term= 0.000001000 term/val=0.002050
g=3 i=0 str=-0.1998 dis= 600000 zq= 1 Rabs=600000 distratio=480 gt2=-3.331e-7 t=-3.331e-7
greenMpole(3): value= 0.000487042 term=-0.000000666 term/val=0.001368
g=4 i=0 str= 0.1998 dis= 800000 zq= 1 Rabs=800000 distratio=640 gt2= 2.498e-7 t= 2.498e-7
greenMpole(4): value= 0.000487541 term= 0.000000500 term/val=0.001025
g=5 i=0 str=-0.1998 dis= 1e+06  zq= 1 Rabs=1e+06  distratio=800 gt2=-1.998e-7 t=-1.998e-7
greenMpole(5): value= 0.000487142 term=-0.000000400 term/val=0.000820
greenPwc(): CollocationGreenPwc(): val=0.000487142
greenPwc(): d=600 r=4800 sp1=(850,1200,0) sp2=(250,1200,0) ---------------------------- (2a)
greenMpole(): spo=(850,1200,0) spc=(250,1200,0) convergenceRadius=1225.77
g=0 i=0 str= 0.1000 dis=0       zq=-1 Rabs=600    distratio=0.49 gt3= 1.230e-4 t= 1.230e-4
g=0 i=1 str= 0.1000 dis=0       zq= 1 Rabs=600    distratio=0.49 gt3= 1.230e-4 t= 2.461e-4
greenMpole(0): value= 0.000246057 term= 0.000246057 term/val=1.000000
g=1 i=0 str=-0.1999 dis= 200000 zq= 1 Rabs=200001 distratio= 163 gt2=-9.997e-7 t=-9.998e-7
greenMpole(1): value= 0.000244057 term=-0.000002000 term/val=0.008193
greenMpole(2): value= 0.000245057 term= 0.000001000 term/val=0.004079
greenMpole(3): value= 0.000244390 term=-0.000000666 term/val=0.002726
greenMpole(4): value= 0.000244890 term= 0.000000500 term/val=0.002040
greenMpole(5): value= 0.000244490 term=-0.000000400 term/val=0.001634
greenMpole(6): value= 0.000244823 term= 0.000000333 term/val=0.001360
greenMpole(7): value= 0.000244538 term=-0.000000285 term/val=0.001167
greenMpole(8): value= 0.000244788 term= 0.000000250 term/val=0.001020
greenMpole(9): value= 0.000244566 term=-0.000000222 term/val=0.000907
greenPwc(): CollocationGreenPwc(): val=0.000244566
greenMpole(): spo=(250,1200,0) spc=(850,1200,0) convergenceRadius=1250 ----------------- (2b)
g=0 i=0 str= 0.1000 dis=0       zq=-1 Rabs=600    distratio=0.48 gt3= 1.260e-4 t= 1.260e-4
g=0 i=1 str= 0.1000 dis=0       zq= 1 Rabs=600    distratio=0.48 gt3= 1.260e-4 t= 2.519e-4
greenMpole(0): value= 0.000251914 term= 0.000251914 term/val=1.000000
g=1 i=0 str=-0.1999 dis= 200000 zq= 1 Rabs=200001 distratio=160  gt2=-9.998e-7 t=-9.998e-7
greenMpole(1): value= 0.000249914 term=-0.000002000 term/val=0.008001
greenMpole(2): value= 0.000250914 term= 0.000001000 term/val=0.003984
greenMpole(3): value= 0.000250248 term=-0.000000666 term/val=0.002662
greenMpole(4): value= 0.000250747 term= 0.000000500 term/val=0.001992
greenMpole(5): value= 0.000250348 term=-0.000000400 term/val=0.001596
greenMpole(6): value= 0.000250681 term= 0.000000333 term/val=0.001328
greenMpole(7): value= 0.000250395 term=-0.000000285 term/val=0.001139
greenMpole(8): value= 0.000250645 term= 0.000000250 term/val=0.000996
greenPwc(): CollocationGreenPwc(2): val=0.000247605
greenPwc(): d=0 r=4800 sp1=(250,1200,0) sp2=(250,1200,0) ------------------------------ (3)
greenMpole(): spo=(250,1200,0) spc=(250,1200,0) convergenceRadius=1225.77
g=0 i=0 str= 0.1000 dis=0       zq=-1 Rabs=0      distratio=0   gt3= 2.721e-4 t= 2.721e-4
g=0 i=1 str= 0.1000 dis=0       zq= 1 Rabs=0      distratio=0   gt3= 2.721e-4 t= 5.442e-4
greenMpole(0): value= 0.000544224 term= 0.000544224 term/val=1.000000
g=1 i=0 str=-0.1999 dis= 200000 zq= 1 Rabs=200000 distratio=163 gt2=-9.998e-7 t=-9.998e-7
greenMpole(1): value= 0.000542225 term=-0.000002000 term/val=0.003688
greenMpole(2): value= 0.000543224 term= 0.000001000 term/val=0.001840
greenMpole(3): value= 0.000542558 term=-0.000000666 term/val=0.001228
greenMpole(4): value= 0.000543058 term= 0.000000500 term/val=0.000920
greenPwc(): CollocationGreenPwc(): val=0.000543058
```

APPENDIX B -- Example: single, -B listing with metalization

```
cap3dInitParam: prePass1=1 optSubRes=1 greenCase=1 greenType=2
greenInit: reading substrate specification (fromParamFile=0)
greenInit: i=1 n=3 m='epi' e=10 b=0
greenInit: i=2 n=3 m='metalization' e=100000 b=-250
-- n=1 bounds=0 matconst=10
-- n=2 bounds=250 matconst=0.1
greenInit: greenData.nlayers[greenCase]=2 maxGreenTerms=500 collocationEps=0.001
--------------------------------------------------------------------------------
greenPwc(): d=0 r=4800 sp1=(600,1200,100000) sp2=(600,1200,100000)
greenMpole(): spo=(600,1200,100000) spc=(600,1200,100000) convergenceRadius=1341.64
g=0 i=0 str= 10      dis= 0    zq=-1 Rabs= 0    distratio=0       gt3= 2.00505e-2 t= 2.00505e-2
g=0 i=1 str=-10      dis= 0    zq= 1 Rabs= 2e+5 distratio=149.071 gt2=-4.99996e-5 t= 2.00005e-2
greenMpole(0): value= 0.020000493 term= 0.020000493 term/val=1.000000
g=1 i=0 str=-9.80198 dis=-2e+5 zq= 1 Rabs= 0    distratio=0       gt3=-1.96535e-2 t=-1.96535e-2
g=1 i=1 str= 9.80198 dis= 2e+5 zq= 1 Rabs= 4e+5 distratio=298.142 gt2= 2.45049e-5 t=-1.96289e-2
greenMpole(1): value= 0.000371545 term=-0.019628948 term/val=52.830645
g=2 i=0 str= 9.60788 dis= 0    zq= 1 Rabs= 2e+5 distratio=149.071 gt2= 4.80390e-5 t= 4.80390e-5
g=2 i=1 str=-9.60788 dis= 4e+5 zq= 1 Rabs= 6e+5 distratio=447.214 gt2=-1.60131e-5 t= 3.20259e-5
greenMpole(2): value= 0.000403571 term= 0.000032026 term/val=0.079356
g=3 i=0 str=-9.41763 dis= 2e+5 zq= 1 Rabs= 4e+5 distratio=298.142 gt2=-2.35440e-5 t=-2.35440e-5
g=3 i=1 str= 9.41763 dis= 6e+5 zq= 1 Rabs= 8e+5 distratio=596.285 gt2= 1.17720e-5 t=-1.17720e-5
greenMpole(3): value= 0.000391799 term=-0.000011772 term/val=0.030046
g=4 i=0 str= 9.23114 dis= 4e+5 zq= 1 Rabs= 6e+5 distratio=447.214 gt2= 1.53852e-5 t= 1.53852e-5
g=4 i=1 str=-9.23114 dis= 8e+5 zq= 1 Rabs=10e+5 distratio=745.356 gt2=-9.23114e-6 t= 6.15408e-6
greenMpole(4): value= 0.000397953 term= 0.000006154 term/val=0.015464
g=5 i=0 str=-9.04834 dis= 6e+5 zq= 1 Rabs= 8e+5 distratio=596.285 gt2=-1.13104e-5 t=-1.13104e-5
g=5 i=1 str= 9.04834 dis=10e+5 zq= 1 Rabs=12e+5 distratio=894.427 gt2= 7.54029e-6 t=-3.77014e-6
greenMpole(5): value= 0.000394183 term=-0.000003770 term/val=0.009564
g=6 i=0 str= 8.86917 dis= 8e+5 zq= 1 Rabs=10e+5 distratio=745.356 gt2= 8.86917e-6 t= 8.86917e-6
g=6 i=1 str=-8.86917 dis=12e+5 zq= 1 Rabs=14e+5 distratio=1043.5  gt2=-6.33512e-6 t= 2.53405e-6
greenMpole(6): value= 0.000396717 term= 0.000002534 term/val=0.006388
g=7 i=0 str=-8.69354 dis=10e+5 zq= 1 Rabs=12e+5 distratio=894.427 gt2=-7.24462e-6 t=-7.24462e-6
g=7 i=1 str= 8.69354 dis=14e+5 zq= 1 Rabs=16e+5 distratio=1192.57 gt2= 5.43346e-6 t=-1.81115e-6
greenMpole(7): value= 0.000394906 term=-0.000001811 term/val=0.004586
g=8 i=0 str= 8.52139 dis=12e+5 zq= 1 Rabs=14e+5 distratio=1043.5  gt2= 6.08671e-6 t= 6.08671e-6
g=8 i=1 str=-8.52139 dis=16e+5 zq= 1 Rabs=18e+5 distratio=1341.64 gt2=-4.73411e-6 t= 1.35260e-6
greenMpole(8): value= 0.000396258 term= 0.000001353 term/val=0.003413
g=9 i=0 str=-8.35265 dis=14e+5 zq= 1 Rabs=16e+5 distratio=1192.57 gt2=-5.22041e-6 t=-5.22041e-6
g=9 i=1 str= 8.35265 dis=18e+5 zq= 1 Rabs=20e+5 distratio=1490.71 gt2= 4.17633e-6 t=-1.04408e-6
greenMpole(9): value= 0.000395214 term=-0.000001044 term/val=0.002642
g=10 i=0 str= 8.1872 dis=16e+5 zq= 1 Rabs=18e+5 distratio=1341.64 gt2= 4.54847e-6 t= 4.54847e-6
g=10 i=1 str=-8.1872 dis=20e+5 zq= 1 Rabs=22e+5 distratio=1639.78 gt2=-3.72148e-6 t= 8.26995e-7
greenMpole(10): value= 0.000396041 term= 0.000000827 term/val=0.002088
g=11 i=0 str=-8.0251 dis=18e+5 zq= 1 Rabs=20e+5 distratio=1490.71 gt2=-4.01256e-6 t=-4.01256e-6
g=11 i=1 str= 8.0251 dis=22e+5 zq= 1 Rabs=24e+5 distratio=1788.85 gt2= 3.34380e-6 t=-6.68761e-7
greenMpole(11): value= 0.000395372 term=-0.000000669 term/val=0.001691
g=12 i=0 str= 7.8662 dis=20e+5 zq= 1 Rabs=22e+5 distratio=1639.78 gt2= 3.57555e-6 t= 3.57555e-6
g=12 i=1 str=-7.8662 dis=24e+5 zq= 1 Rabs=26e+5 distratio=1937.93 gt2=-3.02547e-6 t= 5.50085e-7
greenMpole(12): value= 0.000395922 term= 0.000000550 term/val=0.001389
g=13 i=0 str=-7.7104 dis=22e+5 zq= 1 Rabs=24e+5 distratio=1788.85 gt2=-3.21269e-6 t=-3.21269e-6
g=13 i=1 str= 7.7104 dis=26e+5 zq= 1 Rabs=28e+5 distratio=2087    gt2= 2.75373e-6 t=-4.58955e-7
greenMpole(13): value= 0.000395463 term=-0.000000459 term/val=0.001161
g=14 i=0 str= 7.5577 dis=24e+5 zq= 1 Rabs=26e+5 distratio=1937.93 gt2= 2.90683e-6 t= 2.90683e-6
g=14 i=1 str=-7.5577 dis=28e+5 zq= 1 Rabs=30e+5 distratio=2236.07 gt2=-2.51926e-6 t= 3.87578e-7
greenMpole(14): value= 0.000395851 term= 0.000000388 term/val=0.000979
greenPwc(): CollocationGreenPwc(): val=0.000395657
--------------------------------------------------------------------------------
```

```
greenPwc(): d=0 r=4800 sp1=(600,1200,100000) sp2=(600,1200,100000)
greenMpole(): spo=(600,1200,100000) spc=(600,1200,100000) convergenceRadius=1341.64
g=0 i=0 str= 0.1     dis= 0    zq=-1 Rabs= 0    distratio=0       gt3= 2.00505e-4 t= 2.00505e-4
g=0 i=1 str=-0.1     dis= 0    zq= 1 Rabs= 2e+5 distratio=149.071 gt2=-4.99996e-7 t= 2.00005e-4
greenMpole(0): value= 0.000200005 term= 0.000200005 term/val=1.000000
g=1 i=0 str=-0.09802 dis=-2e+5 zq=-1 Rabs= 2e+5 distratio=149.071 gt1=-4.90095e-7 t=-4.90095e-7
g=1 i=1 str=-0.09802 dis= 2e+5 zq=-1 Rabs= 2e+5 distratio=149.071 gt1=-4.90095e-7 t=-9.80191e-7
g=1 i=2 str= 0.09802 dis=-2e+5 zq= 1 Rabs= 0    distratio=0       gt3= 1.96535e-4 t= 1.95554e-4
g=1 i=3 str= 0.09802 dis= 2e+5 zq= 1 Rabs= 4e+5 distratio=298.142 gt2= 2.45049e-7 t= 1.95799e-4
greenMpole(1): value= 0.000395804 term= 0.000195799 term/val=0.494687
g=2 i=0 str= 0.09608 dis=-4e+5 zq=-1 Rabs= 4e+5 distratio=298.142 gt1= 2.40197e-7 t= 2.40197e-7
g=2 i=1 str= 0.09608 dis= 4e+5 zq=-1 Rabs= 4e+5 distratio=298.142 gt1= 2.40197e-7 t= 4.80393e-7
g=2 i=2 str=-0.09608 dis=-4e+5 zq= 1 Rabs= 2e+5 distratio=149.071 gt2=-4.80390e-7 t= 2.7022e-12
g=2 i=3 str=-0.09608 dis= 4e+5 zq= 1 Rabs= 6e+5 distratio=447.214 gt2=-1.60131e-7 t=-1.60129e-7
greenMpole(2): value= 0.000395644 term=-0.000000160 term/val=0.000405
greenPwc(): CollocationGreenPwc(): val=0.000395724
```

APPENDIX C -- Example: single, -C3 listing

```
cap3dInitParam: prePass1=0 optSubRes=0 greenCase=0 greenType=0
greenInit: reading dielectric specification (fromParamFile=0)
greenInit: i=1 n=3 m='SiO2' e=3.9   b=0
greenInit: i=2 n=3 m='air'  e=0.001 b=250
greenInit: greenData.nlayers[greenCase]=2 maxGreenTerms=500 collocationEps=0.001
greenPwc(): d=0 r=4800 sp1=(850,1200,100000) sp2=(850,1200,100000) ---------------------- (1)
greenMpole(): spo=(850,1200,100000) spc=(850,1200,100000) convergenceRadius=1250
g=0 i=0 str= 1000 dis=0 zq=-1 Rabs=0      distratio=0   gt3= 2.44354    t=2.44354
g=0 i=1 str=-1000 dis=0 zq= 1 Rabs=200000 distratio=160 gt2=-0.00499997 t=2.43854
greenMpole(0): value= 2.438540602 term= 2.438540602 term/val=1.000000
g=1 i=0 str=-999.487 dis=-200000 zq= 1 Rabs=0      distratio=0   gt3=-2.44229    t=-2.44229
g=1 i=1 str= 999.487 dis= 200000 zq= 1 Rabs=400000 distratio=320 gt2= 0.00249871 t=-2.43979
greenMpole(1): value=-0.001248477 term=-2.439789079 term/val=1954.212691
g=2 i=0 str= 998.975 dis= 0      zq= 1 Rabs=200000 distratio=160 gt2= 0.00499484 t=0.00499484
g=2 i=1 str=-998.975 dis= 400000 zq= 1 Rabs=600000 distratio=480 gt2=-0.00166496 t=0.00332988
greenMpole(2): value= 0.002081408 term= 0.003329885 term/val=1.599823
g=3 i=0 str=-998.463 dis= 200000 zq= 1 Rabs=400000 distratio=320 gt2=-0.00249615 t=-0.00249615
g=3 i=1 str= 998.463 dis= 600000 zq= 1 Rabs=800000 distratio=640 gt2= 0.00124808 t=-0.00124807
greenMpole(3):  value= 0.000833333 term=-0.001248075 term/val=1.497690
greenMpole(4):  value= 0.001498633 term= 0.000665300 term/val=0.443938
greenMpole(5):  value= 0.001083034 term=-0.000415599 term/val=0.383736
greenMpole(6):  value= 0.001367870 term= 0.000284836 term/val=0.208233
greenMpole(7):  value= 0.001160283 term=-0.000207587 term/val=0.178910
   ...
greenMpole(87): value= 0.001248599 term=-0.000001264 term/val=0.001012
g=88 i=0 str= 955.875 dis=1.72e+07 zq= 1 Rabs=1.74e+07 distratio=13920 gt2= 5.49353e-05 t=5.4935e-05
g=88 i=1 str=-955.875 dis=1.76e+07 zq= 1 Rabs=1.78e+07 distratio=14240 gt2=-5.37008e-05 t=1.2345e-06
greenMpole(88): value= 0.001249833 term= 0.000001235 term/val=0.000988
greenPwc(): CollocationGreenPwc(): val=0.00124983
greenPwc(): d=600 r=4800 sp1=(850,1200,100000) sp2=(250,1200,100000) ------------------ (2a)
greenMpole(): spo=(850,1200,100000) spc=(250,1200,100000) convergenceRadius=1225.77
g=0 i=0 str= 1000 dis=0 zq=-1 Rabs=600    distratio=0.48949 gt3= 1.23028    t=1.23028
g=0 i=1 str=-1000 dis=0 zq= 1 Rabs=200001 distratio=163.164 gt2=-0.00499995 t=1.22528
greenMpole(0): value= 1.225283582 term= 1.225283582 term/val=1.000000
g=1 i=0 str=-999.487 dis=-200000 Rabs=600    distratio=0.48949 gt3=-1.22965   t=-1.22965
g=1 i=1 str= 999.487 dis= 200000 Rabs=400000 distratio=326.327 gt2= 0.0024987 t=-1.22715
greenMpole(1): value=-0.001870482 term=-1.227154064 term/val=656.063107
g=2 i=0 str= 998.975 dis= 0      Rabs=200001 distratio=163.164 gt2= 0.0049948 t=0.0049948
g=2 i=1 str=-998.975 dis= 400000 Rabs=600000 distratio=489.49  gt2=-0.0016650 t=0.0033298
greenMpole(2):  value= 0.001459383 term= 0.003329865 term/val=2.281694
greenMpole(3):  value= 0.000211310 term=-0.001248073 term/val=5.906351
greenMpole(4):  value= 0.000876609 term= 0.000665299 term/val=0.758946
greenMpole(5):  value= 0.000461010 term=-0.000415599 term/val=0.901497
greenMpole(6):  value= 0.000745846 term= 0.000284836 term/val=0.381897
greenMpole(7):  value= 0.000538260 term=-0.000207587 term/val=0.385663
```

```
   ...
greenMpole(121): value= 0.000626881 term=-0.000000642 term/val=0.001024
greenMpole(122): value= 0.000627513 term= 0.000000631 term/val=0.001006
greenMpole(123): value= 0.000626892 term=-0.000000621 term/val=0.000990
greenPwc(): CollocationGreenPwc(): val=0.000626892
greenMpole(): spo=(250,1200,100000) spc=(850,1200,100000) convergenceRadius=1250 ------ (2b)
g=0 i=0 str= 1000 dis=0 zq=-1 Rabs=600    distratio=0.48    gt3= 1.25957    t=1.25957
g=0 i=1 str=-1000 dis=0 zq= 1 Rabs=200001 distratio=160.001 gt2=-0.00499994 t=1.25457
greenMpole(0): value= 1.254569203 term= 1.254569203 term/val=1.000000
g=1 i=0 str=-999.487 dis=-200000 zq= 1 Rabs=600    distratio=0.48 gt3=-1.25892  t=-1.25892
g=1 i=1 str= 999.487 dis= 200000 zq= 1 Rabs=400000 distratio=320  gt2=0.0024987 t=-1.25642
greenMpole(1): value=-0.001855466 term=-1.256424669 term/val=677.147687
g=2 i=0 str= 998.975 dis=0       Rabs=200001  distratio=160  gt2= 0.00499482 t=0.00499482
g=2 i=1 str=-998.975 dis=400000  Rabs=600000  distratio=480  gt2=-0.00166496 t=0.00332986
greenMpole(2): value= 0.001474397 term= 0.003329863 term/val=2.258458
greenMpole(3): value= 0.000226325 term=-0.001248072 term/val=5.514521
greenMpole(4): value= 0.000891624 term= 0.000665299 term/val=0.746166
greenMpole(5): value= 0.000476025 term=-0.000415599 term/val=0.873062
greenMpole(6): value= 0.000760861 term= 0.000284836 term/val=0.374360
greenMpole(7): value= 0.000553274 term=-0.000207587 term/val=0.375197
   ...
greenMpole(120): value= 0.000642538 term= 0.000000653 term/val=0.001016
greenMpole(121): value= 0.000641896 term=-0.000000642 term/val=0.001000
greenMpole(122): value= 0.000642527 term= 0.000000631 term/val=0.000982
greenPwc(): CollocationGreenPwc(2): val=0.000634709
greenPwc(): d=0 r=4800 sp1=(250,1200,100000) sp2=(250,1200,100000) ---------------------- (3)
greenMpole(): spo=(250,1200,100000) spc=(250,1200,100000) convergenceRadius=1225.77
g=0 i=0 str= 1000 dis=0 zq=-1 Rabs=0      distratio=0       gt3= 2.72112    t=2.72112
g=0 i=1 str=-1000 dis=0 zq= 1 Rabs=200000 distratio=163.163 gt2=-0.00499997 t=2.71612
greenMpole(0):  value= 2.716121090 term= 2.716121090 term/val=1.000000
g=1 i=0 str=-999.487 dis=-200000 zq= 1 Rabs=0      distratio=0      gt3=-2.71973   t=-2.71973
g=1 i=1 str= 999.487 dis= 200000 zq= 1 Rabs=400000 distratio=326.32 gt2= 0.0024987 t=-2.71723
greenMpole(1):  value=-0.001106165 term=-2.717227255 term/val=2456.438593
g=2 i=0 str= 998.975 dis= 0      zq= 1 Rabs=200000 distratio=163.16 gt2= 0.0049948 t=0.00499484
g=2 i=1 str=-998.975 dis= 400000 zq= 1 Rabs=600000 distratio=489.49 gt2=-0.0016649 t=0.00332989
greenMpole(2):  value= 0.002223721 term= 0.003329886 term/val=1.497439
greenMpole(3):  value= 0.000975646 term=-0.001248075 term/val=1.279230
greenMpole(4):  value= 0.001640945 term= 0.000665300 term/val=0.405437
greenMpole(5):  value= 0.001225346 term=-0.000415599 term/val=0.339169
greenMpole(6):  value= 0.001510183 term= 0.000284836 term/val=0.188611
greenMpole(7):  value= 0.001302596 term=-0.000207587 term/val=0.159364
   ...
greenMpole(83): value= 0.001390849 term=-0.000001391 term/val=0.001000
g=84 i=0 str= 957.838 dis=1.64e+07 Rabs=1.66e+07 distratio=13542.6 gt2= 5.77011e-5 t=5.77011e-5
g=84 i=1 str=-957.838 dis=1.68e+07 Rabs=1.70e+07 distratio=13868.9 gt2=-5.63434e-5 t=1.35767e-6
greenMpole(84): value= 0.001392206 term= 0.000001358 term/val=0.000975
greenPwc(): CollocationGreenPwc(): val=0.00139221
```

APPENDIX D -- greenSeries() for greenType 0 and nLayers=3

```
greenSeries (0, 3, 1, 1) {
    image(-(k(1,1 + p(0) + p(2))*k(2,p(1) + p(2))/e(1)),
        2*t(1)*(-1 - p(0) + p(2)) - 2*t(2)*(p(1) + p(2)),-1);
    image(-(k(1,1 + p(0) + p(2))*k(2,p(1) + p(2))/e(1)),
        2*t(1)*(1 + p(0) - p(2)) + 2*t(2)*(p(1) + p(2)),-1);
    image(k(1,1 + p(0) + p(2))*k(2,p(1) + p(2))/e(1),
        2*t(1)*(-1 - p(0) + p(2)) - 2*t(2)*(p(1) + p(2)),1);
    image(k(1,1 + p(0) + p(2))*k(2,p(1) + p(2))/e(1),
        2*t(1)*(1 + p(0) - p(2)) + 2*t(2)*(p(1) + p(2)),1);
    image(-(k(1,p(0) + p(2))*k(2,1 + p(1) + p(2))/e(1)),
        2*t(1)*(-p(0) + p(2)) - 2*t(2)*(1 + p(1) + p(2)),-1);
    image(-(k(1,p(0) + p(2))*k(2,1 + p(1) + p(2))/e(1)),
        2*t(1)*(p(0) - p(2)) + 2*t(2)*(1 + p(1) + p(2)),-1);
    image(k(1,p(0) + p(2))*k(2,1 + p(1) + p(2))/e(1),
        2*t(1)*(-p(0) + p(2)) - 2*t(2)*(1 + p(1) + p(2)),1);
    image(k(1,p(0) + p(2))*k(2,1 + p(1) + p(2))/e(1),
        2*t(1)*(p(0) - p(2)) + 2*t(2)*(1 + p(1) + p(2)),1);
}

greenSeries (0, 3, 2, 1) { ... }  /* 12 images */

greenSeries (0, 3, 3, 1) { ... }  /* 12 images */

greenSeries (0, 3, 1, 2) { ... }  /* 12 images */

greenSeries (0, 3, 2, 2) {
    image(-(k(1,1 + p(0) + p(2))*k(2,p(1) + p(2))/e(2)),
        2*t(1)*(-1 + p(0) - p(2)) + 2*t(2)*(p(1) + p(2)),1);
    image(k(1,1 + p(0) + p(2))*k(2,p(1) + p(2))/e(2),
        2*t(1)*(1 + p(0) - p(2)) + 2*t(2)*(p(1) + p(2)),1);
    image(-(k(1,p(0) + p(2))*k(2,1 + p(1) + p(2))/e(2)),
        2*t(1)*(-p(0) + p(2)) - 2*t(2)*(1 + p(1) + p(2)),-1);
    image(-(k(1,p(0) + p(2))*k(2,1 + p(1) + p(2))/e(2)),
        2*t(1)*(p(0) - p(2)) + 2*t(2)*(1 + p(1) + p(2)),-1);
    image(k(1,p(0) + p(2))*k(2,1 + p(1) + p(2))/e(2),
        2*t(1)*(-p(0) + p(2)) - 2*t(2)*(1 + p(1) + p(2)),1);
    image(k(1,p(0) + p(2))*k(2,1 + p(1) + p(2))/e(2),
        2*t(1)*(p(0) - p(2)) + 2*t(2)*(1 + p(1) + p(2)),1);
    image(-(k(1,1 + p(0) + p(2))*k(2,1 + p(1) + p(2))/e(2)),
        2*t(1)*(1 - p(0) + p(2)) - 2*t(2)*(1 + p(1) + p(2)),-1);
    image(-(k(1,1 + p(0) + p(2))*k(2,1 + p(1) + p(2))/e(2)),
        2*t(1)*(-1 + p(0) - p(2)) + 2*t(2)*(1 + p(1) + p(2)),-1);
    image(k(1,1 + p(0) + p(2))*k(2,1 + p(1) + p(2))/e(2),
        2*t(1)*(-1 - p(0) + p(2)) - 2*t(2)*(1 + p(1) + p(2)),1);
    image(k(1,1 + p(0) + p(2))*k(2,1 + p(1) + p(2))/e(2),
        2*t(1)*(-1 + p(0) - p(2)) + 2*t(2)*(1 + p(1) + p(2)),1);
}

greenSeries (0, 3, 3, 2) { ... }  /* 10 images */

greenSeries (0, 3, 1, 3) { ... }  /* 12 images */
greenSeries (0, 3, 2, 3) { ... }  /* 10 images */

greenSeries (0, 3, 3, 3) {
    image(-(k(1,1 + p(0) + p(2))*k(2,p(1) + p(2))/e(3)),
        2*t(1)*(-1 + p(0) - p(2)) + 2*t(2)*(p(1) + p(2)),1);
    image(k(1,1 + p(0) + p(2))*k(2,p(1) + p(2))/e(3),
        2*t(1)*(1 + p(0) - p(2)) + 2*t(2)*(p(1) + p(2)),1);
    image(-(k(1,p(0) + p(2))*k(2,1 + p(1) + p(2))/e(3)),
        2*t(1)*(p(0) - p(2)) + 2*t(2)*(-1 + p(1) + p(2)),1);
    image(k(1,p(0) + p(2))*k(2,1 + p(1) + p(2))/e(3),
        2*t(1)*(p(0) - p(2)) + 2*t(2)*(1 + p(1) + p(2)),1);
    image(-(k(1,1 + p(0) + p(2))*k(2,1 + p(1) + p(2))/e(3)),
        2*t(1)*(1 + p(0) - p(2)) + 2*t(2)*(-1 + p(1) + p(2)),1);
    image(k(1,1 + p(0) + p(2))*k(2,1 + p(1) + p(2))/e(3),
        2*t(1)*(-1 + p(0) - p(2)) + 2*t(2)*(1 + p(1) + p(2)),1);
}
```

APPENDIX E -- greenSeries() for greenType 2 and 3

```
/* T.S.:
 * value 2 for first argument denotes substrate resistance Green’s function.
 */
greenSeries (2, 2, 1, 1) {
/* T.S.:
 * this only implements a limited green’s function (z0==0, b==0)
 */
    image(2*k(1,p(0))/s(1),-2*b(1)*p(0), 1);
    image(2*k(1,p(0))/s(1), 2*b(1)*p(0), 1);
}

/* SdeG:
 * value 3 denotes grounded substrate resistance Green’s function.
 */
greenSeries (3, 2, 1, 1) {
    image(-(k(1,1 + p(0))/e(1)),-2*t(1)*(1 + p(0)),-1);
    image(-(k(1,1 + p(0))/e(1)), 2*t(1)*(1 + p(0)),-1);
    image(  k(1,1 + p(0))/e(1) ,-2*t(1)*(1 + p(0)), 1);
    image(  k(1,1 + p(0))/e(1) , 2*t(1)*(1 + p(0)), 1);
}

greenSeries (3, 2, 2, 2) {
    image(-(k(1,1 + p(0))/e(2)),-2*t(1)*(1 - p(0)), 1);
    image(  k(1,1 + p(0))/e(2) , 2*t(1)*(1 + p(0)), 1);
}

greenSeries (3, 3, 2, 2) { ... }  /* 10 images */

greenSeries (3, 3, 3, 3) {
    image(-(k(1,1 + p(0) + p(2))*k(2,p(1) + p(2))/e(3)),
        2*t(1)*(-1 + p(0) - p(2)) + 2*t(2)*(p(1) + p(2)), 1);
    image(  k(1,1 + p(0) + p(2))*k(2,p(1) + p(2))/e(3),
        2*t(1)*(1 + p(0) - p(2)) + 2*t(2)*(p(1) + p(2)), 1);
    image(-(k(1,p(0) + p(2))*k(2,1 + p(1) + p(2))/e(3)),
        2*t(1)*(p(0) - p(2)) + 2*t(2)*(-1 + p(1) + p(2)), 1);
    image(  k(1,p(0) + p(2))*k(2,1 + p(1) + p(2))/e(3),
        2*t(1)*(p(0) - p(2)) + 2*t(2)*(1 + p(1) + p(2)), 1);
    image(-(k(1,1 + p(0) + p(2))*k(2,1 + p(1) + p(2))/e(3)),
        2*t(1)*(1 + p(0) - p(2)) + 2*t(2)*(-1 + p(1) + p(2)), 1);
    image(  k(1,1 + p(0) + p(2))*k(2,1 + p(1) + p(2))/e(3),
        2*t(1)*(-1 + p(0) - p(2)) + 2*t(2)*(1 + p(1) + p(2)), 1);
}
```

APPENDIX F -- Example: single, -C3 listing with nLayers=3

```
greenPwc(): d=0 r=4800 sp1=(600,1200,100000) sp2=(600,1200,100000)
greenMpole(): spo=(600,1200,100000) spc=(600,1200,100000) convergenceRadius=1341.64
greenInit: i=1 n=3 m='SiO2' e=3.9 b=0
greenInit: i=2 n=3 m='SiX'  e=2.0 b=10
greenInit: i=3 n=3 m='air'  e=0.1 b=250 ------------------------------------------
greenMpole( 0): value= 0.020000493 term= 0.020000493 term/val=1.000000
greenMpole( 1): value= 0.001525835 term=-0.018474658 term/val=12.107901
greenMpole( 2): value= 0.001985768 term= 0.000459933 term/val=0.231615
greenMpole( 3): value= 0.001866520 term=-0.000119248 term/val=0.063888
greenMpole( 4): value= 0.001928849 term= 0.000062329 term/val=0.032314
greenMpole( 5): value= 0.001879278 term=-0.000049571 term/val=0.026378
greenMpole( 6): value= 0.001927613 term= 0.000048334 term/val=0.025075
greenMpole( 7): value= 0.001875550 term=-0.000052063 term/val=0.027759
greenMpole( 8): value= 0.001934940 term= 0.000059390 term/val=0.030694
greenMpole( 9): value= 0.001864528 term=-0.000070413 term/val=0.037764
greenMpole(10): value= 0.001950422 term= 0.000085895 term/val=0.044039
greenMpole(11): value= 0.001843283 term=-0.000107139 term/val=0.058124
greenMpole(12): value= 0.001979348 term= 0.000136065 term/val=0.068742
greenMpole(13): value= 0.001803955 term=-0.000175393 term/val=0.097227
greenMpole(14): value= 0.002032900 term= 0.000228945 term/val=0.112620

greenInit: i=3 n=3 m='air' e=0.01 b=250 ------------------------------------------
greenMpole( 0): value= 0.200004931 term= 0.200004931 term/val=1.000000
greenMpole( 1): value=-0.002155667 term=-0.202160597 term/val=93.781017
greenMpole( 2): value= 0.002925938 term= 0.005081604 term/val=1.736744
greenMpole( 3): value= 0.001544615 term=-0.001381323 term/val=0.894283
greenMpole( 4): value= 0.002330906 term= 0.000786291 term/val=0.337333
greenMpole( 5): value= 0.001647387 term=-0.000683519 term/val=0.414911
greenMpole( 6): value= 0.002369832 term= 0.000722444 term/val=0.304851
greenMpole( 7): value= 0.001530545 term=-0.000839287 term/val=0.548358
greenMpole( 8): value= 0.002560979 term= 0.001030434 term/val=0.402359
greenMpole( 9): value= 0.001247251 term=-0.001313729 term/val=1.053300
greenMpole(10): value= 0.002969934 term= 0.001722683 term/val=0.580041
greenMpole(11): value= 0.000660587 term=-0.002309346 term/val=3.495899
greenMpole(12): value= 0.003812240 term= 0.003151653 term/val=0.826719
greenMpole(13): value=-0.000553171 term=-0.004365411 term/val=7.891614
greenMpole(14): value= 0.005569531 term= 0.006122702 term/val=1.099321

greenInit: i=3 n=3 m='air' e=0.001 b=250 -----------------------------------------
greenMpole( 0): value= 2.000049309 term= 2.000049309 term/val=1.000000
greenMpole( 1): value=-0.039832243 term=-2.039881552 term/val=51.211817
greenMpole( 2): value= 0.011494832 term= 0.051327075 term/val=4.465231
greenMpole( 3): value=-0.002526674 term=-0.014021506 term/val=5.549392
greenMpole( 4): value= 0.005525017 term= 0.008051691 term/val=1.457315
greenMpole( 5): value=-0.001537113 term=-0.007062130 term/val=4.594412
greenMpole( 6): value= 0.005987696 term= 0.007524809 term/val=1.256712
greenMpole( 7): value=-0.002820719 term=-0.008808416 term/val=3.122755
greenMpole( 8): value= 0.008074112 term= 0.010894831 term/val=1.349354
greenMpole( 9): value=-0.005918070 term=-0.013992182 term/val=2.364315
greenMpole(10): value= 0.012563968 term= 0.018482038 term/val=1.471035
greenMpole(11): value=-0.012392936 term=-0.024956904 term/val=2.013801
greenMpole(12): value= 0.021914847 term= 0.034307783 term/val=1.565504
greenMpole(13): value=-0.025951378 term=-0.047866225 term/val=1.844458
greenMpole(14): value= 0.041671699 term= 0.067623077 term/val=1.622758
greenMpole(15): value=-0.054890495 term=-0.096562194 term/val=1.759179
greenMpole(16): value= 0.084278712 term= 0.139169207 term/val=1.651297
greenMpole(17): value=-0.117929623 term=-0.202208335 term/val=1.714653
greenMpole(18): value= 0.177981190 term= 0.295910813 term/val=1.662596
greenMpole(19): value=-0.257815636 term=-0.435796826 term/val=1.690343
greenMpole(20): value= 0.387658700 term= 0.645474336 term/val=1.665058
greenMpole(21): value=-0.573288295 term=-0.960946995 term/val=1.676202
greenMpole(22): value= 0.863968912 term= 1.437257207 term/val=1.663552
greenMpole(23): value=-1.294779393 term=-2.158748305 term/val=1.667271
greenMpole(24): value= 1.945729347 term= 3.240508740 term/val=1.665447
greenMpole(25): value=-2.713905154 term=-4.659634501 term/val=1.716948
greenMpole(26): value= 3.617200293 term= 6.331105447 term/val=1.750278
```

APPENDIX G -- Example: single, -B listing with nLayers=3

```
greenInit: i=1 n=3 m='epi'  e=10 b=0
greenInit: i=2 n=3 m='epi2' e=40 b=-215.45
greenInit: i=3 n=3 m='metalization' b=-250
-- n=1 bounds=0     matconst=40
-- n=2 bounds=34.55 matconst=10
-- n=3 bounds=250   matconst=0.1
greenInit: greenData.nlayers[greenCase]=3
greenInit: maxGreenTerms=500
greenInit: collocationEps=0.001
greenPwc(): d=0 r=4800 sp1=(600,1200,100000) sp2=(600,1200,100000)
greenMpole(): spo=(600,1200,100000) spc=(600,1200,100000)
greenMpole(): convergenceRadius=1341.64 multipolesMindist=2
greenMpole( 0): value= 0.000200005 term= 0.000200005 term/val=1.000000
greenMpole( 1): value= 0.000397322 term= 0.000197317 term/val=0.496618
greenMpole( 2): value= 0.000394909 term=-0.000002414 term/val=0.006112
greenMpole( 3): value= 0.000395864 term= 0.000000955 term/val=0.002412
greenMpole( 4): value= 0.000395238 term=-0.000000625 term/val=0.001582
greenMpole( 5): value= 0.000395777 term= 0.000000538 term/val=0.001360
greenMpole( 6): value= 0.000395234 term=-0.000000543 term/val=0.001374
greenMpole( 7): value= 0.000395832 term= 0.000000598 term/val=0.001512
greenMpole( 8): value= 0.000395144 term=-0.000000688 term/val=0.001742
greenMpole( 9): value= 0.000395938 term= 0.000000794 term/val=0.002006
greenMpole(10): value= 0.000395064 term=-0.000000874 term/val=0.002212
greenMpole(11): value= 0.000395888 term= 0.000000824 term/val=0.002080
greenMpole(12): value= 0.000395492 term=-0.000000396 term/val=0.001001
greenMpole(13): value= 0.000394523 term=-0.000000969 term/val=0.002456
greenMpole(14): value= 0.000399025 term= 0.000004502 term/val=0.011281
greenMpole(15): value= 0.000386163 term=-0.000012862 term/val=0.033306
greenMpole(16): value= 0.000417881 term= 0.000031718 term/val=0.075903
greenMpole(17): value= 0.000344831 term=-0.000073051 term/val=0.211845
greenMpole(18): value= 0.000506843 term= 0.000162012 term/val=0.319650
greenMpole(19): value= 0.000155653 term=-0.000351190 term/val=2.256238
greenMpole(20): value= 0.000905845 term= 0.000750192 term/val=0.828168
    ...
greenMpole(117): value=-0.000676771 term=-0.003200493 term/val=4.729062
space3d: No more core.
Already allocated 16475583 bytes, cannot get 1685040 more.
```

CONTENTS