Space Green Module Usage

S. de Graaf

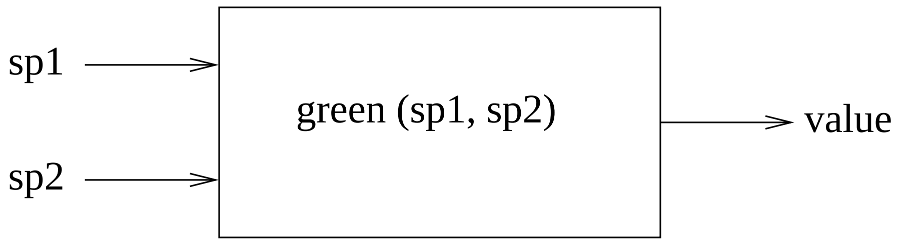
Circuits and Systems Group
Faculty of Electrical Engineering,
Mathematics and Computer Science
Delft University of Technology
The Netherlands

Report EWI-ENS 10-03
June 21, 2010

Last revision: June 24, 2010.

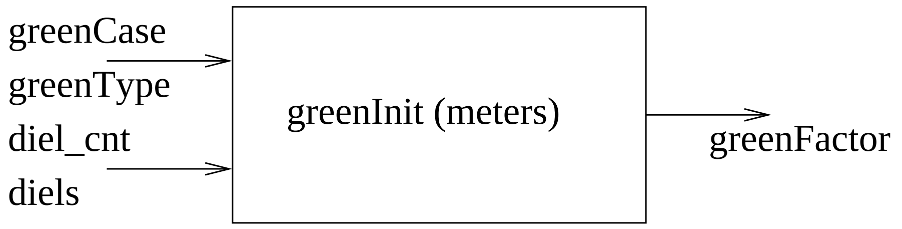
1. INTRODUCTION

Because the Green module is a blackbox for us, we want to know more of it. We know that two spiders must be given to it and that a Green value is calculated for this two. See also the "Space Green API Implementation Note" in "share/doc/an0202.pdf".



Besides that the spiders needs to have coordinates, the spiders needs also be connected to a face or an edge data structure depending on operation mode.

Before we can use function green, we need to init the Green module. This must be done with a call to function greenInit. This function needs some global variables and data to work with. It has only one argument "meters" which specifies the unit (scaling factor) for the spider coordinates. Function greenInit returns the greenFactor, which is equal to "meters * 4 * PI". This factor can be used to scale the resulting Green values. Besides that the constant Epsilon0 is defined (8.855e-12 F/m), which is used to calculate the capacitance values.



For CAP3D mode the greenCase and greenType must be set to DIEL (= 0). A global array of dielectric medium data must exist. The number of dielectric layers is specified with "diel_cnt". Normally these data comes from a space technology file. Note that not more than three layers can be specified and the values of the bottoms must be given in microns. With the *tecc* technology compiler, however, it is a possible to use more layers by using the unigreen method. Then a special technology file is produced with calculated Green data, but this cost a lot of time to produce.
Function greenInit shall also init a lot of Green parameters, which default values can be changed by specifying these parameters in a space parameter file (or on the command line). For CAP3D, most of these parameters starts with "cap3d.xxx", where "xxx" is the name of one of the parameters.

2. WHICH DATA USES THE GREEN MODULE

It is important to know which data must be given to the Green module. Nornally the spiders are generated on base of a 3D layout boundary mesh. The faces are the top, bottom and sidewall faces of the 3D conductors. The edges are the edges of the faces. Depending on the chosen mode the data must be filled in correctly.

2.1 SPIDER DATA

The following spider members are used by the Green module:

```
spider -> act_x
spider -> act_y
spider -> act_z
spider -> face
spider -> edge
spider -> moments
```

First of all, the spider actual x,y,z coordinates. Second, if the spider is a center spider of a face (PWC mode), then the spider must point to a face data structure and not point to a edge data structure (edge = NULL). If the spider is an edge spider of a face (PWL mode), then the spider must not point to a face data structure (face = NULL), but it must point to a face edge data structure. Third, the spider moments member must be inited to NULL. This member is set and used by the Green multipole method.

2.2 FACE DATA

The following face members are used by the Green module:

```
face -> corners[0]
face -> corners[1]
face -> corners[2]
face -> corners[3]
face -> area
face -> len
```

First of all, the face has three or four corner spiders. The face corners array points to the corner spiders. Member corners[3] is NULL when the face has only three corners. For each face the members "area" and "len" must be calculated, the values can not be zero. The "len" of a face is the maximum found edge length of that face. This two values can be calculated from the corner points of the face.

3. WHICH MODES USES THE GREEN MODULE

The modes used are dependend of the parameter settings.

3.1 PWC VERSUS PWL

Function green calls eighter function greenPwl or greenPwc depending on the setting of variable FeModePwl. This is the main entrance of the Green module:

```
double green (spider_t *sp1, spider_t *sp2)
{
    val = FeModePwl ? greenPwl (sp1, sp2) : greenPwc (sp1, sp2);
    if (val < 0) val = 0;
    return (val);
}
```

As you see, function green cannot return a negative value. Thus, the choice is between Piecewise Linear (PWL) or Piecewise Constant (PWC). Only for be_mode "1c" and "1g" variable FeModePwl is TRUE. This is set by function greenInit.

3.2 PWC MODE

Piecewise Constant (PWC) collocation (be_mode "0c") is the default Green mode. Piecewise Constant galerkin mode can be set with "0g". In that case variable FeModeGalerkin becomes TRUE. By Piecewise Constant mode the center spiders of the faces must be given to the Green module. The face "len" is used for making a function choice in combination with the distance of the spiders. The face "area" is used by CollocationGreenPwc and GalerkinGreenPwc, but only when the multipole method is not used.

```
Private green_t greenPwc (spider_t *sp1, spider_t *sp2)
{
    r = sp1 -> face -> len + sp2 -> face -> len;  /* r > 0 */
    d = spiderDist (sp1, sp2);  /* d >= 0 */

    if (d >= r * pointGreenDist && d > 0) {
      val = PointGreen (sp1, sp2);
    }
    else if (d >= r * collocationGreenDist) {
        val = CollocationGreenPwc (sp1, sp2);
      if (d < r * asymCollocationGreenDist && sp1 != sp2) {
          val += CollocationGreenPwc (sp2, sp1);
          val /= 2;
      }
    }
    else {
      val = GalerkinGreenPwc (sp1, sp2);
    }
    return (val);
}
```

Variable pointGreenDist is default "infinity", thus function PointGreen is normally not called. This variable can be set with parameter "cap3d.point_green". Note that PointGreen is not called when spider sp1 is equal to spider sp2.
Variable collocationGreenDist is default "0", thus function CollocationGreenPwc is always called. This variable can be set with parameter "cap3d.collocation_green", then CollocationGreenPwc is only called for d >= r*collocationGreenDist, else function GalerkinGreenPwc is called. Note that for equal spiders (d = 0) CollocationGreenPwc is called only for collocationGreenDist is "0". When FeModeGalerkin is TRUE, then collocationGreenDist is default "infinity" and GalerkinGreenPwc is always called.
Function CollocationGreenPwc is always called twice when spider sp1 is not equal to spider sp2 and variable asymCollocationGreenDist is "infinity". This variable can be set with parameter "cap3d.asym_collocation_green".

Note that both functions CollocationGreenPwc and GalerkinGreenPwc shall default call function greenMpole and return directly (but only if greenMpole returns TRUE). This because parameter "use_multipoles" is default "on" and useMultiPoles is TRUE.

```
Private green_t CollocationGreenPwc (spider_t *spo, spider_t *spc)
{
    if (useMultiPoles && greenMpole (spo, spc, &val)) return val;

    area = spc -> face -> area;
    c = spc -> face -> corners;
    R3Set (opp, spo);
    R3Set (cp1, c[0]);
    R3Set (cp2, c[1]);
    R3Set (cp3, c[2]);
    R3Set (cp4, (c[3] ? c[3] : c[0])); /* triangle or quadrilateral */
    val = doCollocationGreen (opp, cp1, cp2, cp3, cp4);
    return (val / (area * y_stretch));
}
```

You see above that CollocationGreenPwc uses the face "area" and variable "y_stretch" and also the face corner spiders (this is also the case for function GalerkinGreenPwc). Variable "y_stretch" is default "1" but can be changed with parameter "cap3d.y_stretch". Note that this variable is also used by R3Set for the y-coordinate.

3.3 MULTIPOLE MODE

Default, for both collocation and galerkin, the multipole function greenMpole is called (parameter "use_multipoles" is "on"). For CollocationGreenPwc with unequal spiders it is two times called, because the mean value is used. Function greenMpole internally uses variable FeModeGalerkin to decide its working. It calls function getPiePoints, which uses the face corner spiders for PWC (the face area is not used). For PWL however, the spider edge is used and if the edge points to a face, then the code tries to go counter-clockwise (using macro ccwa) around the face. Note that in PWL mode triangular faces are used, which is checked by getPiePoints. Note that the Green module is not completely self supporting, because macro ccwa calls function meshCcwAdjacent from the spider module. Function getPiePoints calls function mediumNumber2 to test of all peripheral points are laying in the same dielectric medium. If one point is crossing a dielectric interface, then FALSE is returned. In that case function greenMpole is returning FALSE and another method must be used.

The multipole method uses some parameters for fine tuning.
For example:

```
    multipolesMindist = paramLookupD ("cap3d.mp_min_dist", "2.0");

    oMaxMpOrder = cMaxMpOrder = paramLookupI ("cap3d.mp_max_order", "2");
    defval = mprintf ("%d", oMaxMpOrder);
    oMaxMpOrder = paramLookupI ("multipoles_oMaxMpOrder", defval);
    cMaxMpOrder = paramLookupI ("multipoles_cMaxMpOrder", defval);
```

3.4 PWL MODE

Piecewise Linear mode can be set with be_mode "1c" or "1g". Thus variable FeModePwl becomes TRUE. By "1g" galerkin is used and variable FeModeGalerkin becomes TRUE. By Piecewise Linear mode the edge spiders of the faces must be given to the Green module (no center spiders are used). The edges must point to faces, which must have a triangular shape (three corner points). The face "len" is not used in this mode. The face "area" is used by CollocationGreenPwl and GalerkinGreenPwl, but only when the multipole method is not used.

```
Private green_t CollocationGreenPwl (spider_t *spo, spider_t *spc)
{
    if (useMultiPoles && greenMpole (spo, spc, &val)) return (val);

    area = val = 0;
    R3Set (opp, spo);
    for (e = spc -> edge; e; e = NEXT_EDGE (spc, e)) {
        if (!(f = e -> face)) continue; /* sheet conductor */

        sp2 = ccwa (spc, f);
        sp3 = ccwa (sp2, f);
        ASSERT (sp2 != spc && sp2 != sp3 && sp3 != spc);

        R3Set (cp1, spc);
        R3Set (cp2, sp2);
        R3Set (cp3, sp3);
        val  += doCollocationGreen (opp, cp1, cp2, cp3, cp1);
        area += f -> area;
    }
    return (val / (area * y_stretch));
}
```

Note that the "area" must be set unequal zero, thus there must be a face "f".

4. SOME OTHER FUNCTIONS

4.1 ccwa

This code is only used for Piecewise Linear mode.

```
#define ccwa(sp, f) meshCcwAdjacent (sp, f, (face_t *) NULL)

/* Find next spider in ccw (counter-clock wise) order around face.
 */
spider_t *meshCcwAdjacent (spider_t *sp, face_t *face, face_t *newface)
{
    spiderEdge_t *e;

    for (e = sp -> edge; e && e -> face != face; e = NEXT_EDGE (sp, e));

    if (e) {
        if (newface) e -> face = newface;
        return (OTHER_SPIDER (sp, e));
    }
    say ("Internal error %s:%d", __FILE__, __LINE__); die ();
    return (NULL);
}
```

4.2 NEXT_EDGE

```
#define NEXT_EDGE(sp, e)  (e -> nb)
```

4.3 OTHER_SPIDER

```
#define OTHER_SPIDER(sp, e)  (e -> oh -> sp)
```

4.4 PointGreen

Function PointGreen can only be called when parameter "cap3d.point_green" is set less than infinity. Note that the face corner coordinates are not used.

```
Private green_t PointGreen (spider_t *sp1, spider_t *sp2)
{
    pointR3_t p1, p2;

    R3Set (p1, sp1);
    R3Set (p2, sp2);
    return (greenFunc (&p1, &p2));
}
```

4.5 R3Set

This macro places the spider coordinates into a pointR3_t data structure.

```
#define R3Set(p, sp) \
        R3Assign(p, sp->act_x, sp->act_y * y_stretch, sp->act_z)

#define R3Assign(p, a, b, c)   (p.x = a, p.y = b, p.z = c)
```

5. OVERVIEW OF THE GREEN PARAMETERS

5.1 Standard Green Parameters

`cap3d.be_mode (string) (default: "0c" or "collocation")`

Other values are: "0g" (or "galerkin"), "1c" and "1g".
See the space 3D capacitance extraction manual.

`cap3d.point_green (real > 0) (default: infinity)`

Sets variable pointGreenDist. See this note.
When set, function PointGreen is called by d >= pointGreenDist.

`cap3d.collocation_green (real) (default: FeModeGalerkin ? infinity : 0)`

Sets variable collocationGreenDist. See code in this note.
When set, function CollocationGreenPwc/l is called by d >= collocationGreenDist.
When d < collocationGreenDist function GalerkinGreenPwc/l is called instead.
Note that pointGreenDist has higher priority (if set).

`cap3d.asym_collocation_green (real) (default: infinity)`

Sets variable asymCollocationGreenDist. See code in this note.
When function CollocationGreenPwc/l is called, it is called twice for unequal spiders and the mean value is used, but only when d < asymCollocationGreenDist.

`cap3d.max_green_terms (integer >= 1) (default: 500)`

See the space 3D capacitance extraction manual.

`cap3d.green_eps (real > 0) (default: 0.001)`

See the space 3D capacitance extraction manual.

`cap3d.col_rel_eps (real > 0) (default: 0.2)`

In Galerkin mode, the "cap3d.green_eps" value multiplied with "cap3d.col_rel_eps" is used. This is done to get a higher accuracy of collocation.

5.2 Debug Parameters

`debug.print_green_init (bool) (default: off)`

Prints greenInit setup information. Which Green mode is used (be_mode, FeModeGalerkin, FeModePwl) The dielectric data which is used and some other settings.

`debug.print_green_terms (bool) (default: off)`

Prints the green terms.

`debug.print_green_gterms (bool) (default: off)`

Prints all the green terms.

5.3 Multipole Parameters

```
use_multipoles (bool) (default: on)
```

```
force_mp_anaint (bool) (default: on)
```

When TRUE, enforce use of analytical formulas for inner (collocation) integral in multipole routine for Green function.

```
force_mp_exint (bool) (default: off)
```

When TRUE, enforce use of analytical and adaptive numerical integration in multipole routine for Green function.

```
cap3d.mp_min_dist (real) (default: 2.0)
```

See space 3D capacitance extraction manual.

```
cap3d.mp_max_order (integer 0 ... 3) (default: 2)
```

See space 3D capacitance extraction manual.

```
multipoles_cMaxMpOrder (default: cap3d.mp_max_order)
multipoles_oMaxMpOrder (default: cap3d.mp_max_order)
```

See parameter "cap3d.mp_max_order".

5.4 Unigreen Parameters

```
cap3d.use_unigreen (bool) (default: diel_blob_environment ? on : off)
```

When there is no diel_blob_environment, it is put off. A diel_blob_environment exists, when there is a non empty unigreen technology file.

```
cap3d.use_unigreen_interpolation (bool) (default: on)
```

5.5 Integration Parameters

```
use_adptv_intgrtn (bool) (default: off)
```

When TRUE, use adaptive integration when Stroud’s formulas failed to achieve the desired precision, in conventional calculation of Green function (galerkin).

```
force_adptv_intgrtn (bool) (default: off)
```

When TRUE, enforce use of adaptive integration instead of Stroud’s formulas, in conventional calculation of Green function to achieve the desired precision (galerkin).

5.6 doCollocationGreen Parameters

```
an_turnover (integer > 0) (default: 100)
```

This value is used as argument to functions integrate1D and integrate2D. This is the maximum number of refinement steps that is tried in sequence to reach the target accuracy. It can be used to control running time. For galerkin a fixed an_turnover of "1000" is used.

`collocation_mode (integer) (default: 0)`

When "1", then numeric collocation is not done for off diagonal images.

`accelerate_levin (bool) (default: off)`

When "on", then "accelerate_levin" is done for numeric collocation.

5.7 Special Green Parameters

`cap3d.use_mean_green_values (bool) (default: off)`

See the Green API note.

`cap3d.merge_images (bool) (default: on)`

See the Green API note.

`cap3d.use_lowest_medium (bool) (default: on)`

See the Green API note.

`cap3d.use_old_images (bool) (default: off)`

See the Green API note.

`cap3d.y_stretch (real > 0) (default: 1.0)`

When set, a spider y-coordinate correction is applied. See code in this note.