1 VGFIN

1.1 Original description:

Program to set up a particle dipole array for the VGF code. A particle (sphere, oblate or prolate spheroid) is split into a cubic array of dipole cells (cubic cells with a dipole at the center). A course array of NSID^3 dipole cells is defined using a cubic lattice. The particle is, in effect, carved out of a NSID^3 cube of dipole cells. The course array cells are divided into NLSID fine array cells. Each of the fine array cells is tested to see if it is within the particle geometry. If so, it is counted. The course array cell is weighted in volume (buy adjusting the cell length on a side, (D) according to how many of it's fine array cells are in the particle.

1.2 Units

Angles are input in degrees and converted to radians. Lengths are all relative, that is, if you input a wavelength of 10 um then all other lengths must be in um. You may use m or cm or furlongs if you wish as long as all lengths are in the same units.

1.3 Time Dependence

The VGF code uses the time dependence exp(-iwt). This code (VGFIN) receives the index of refraction from the user and modifies the sign of the imaginary part to fit the VGF time dependence convention (i.e. the imaginary part should be positive).

1.4 Geometry

The particle is aligned with it's symmetry axis along the Z-axis. The incident plane wave may be rotated

around the particle by the angles alpha, beta, and gamma (corresponding to the Euler angles as given in Arfkin, Mathematical Methods for Physicists, 2nd Ed., Academic Press, NY, 1970). Gamma is effectively the polarization angle around the incident (k-hat) direction and is input as such.

1.5 Variable Definitions

1.5.1 Initialized variables

The code initializes variables in lines 46-66. ForTran requires variables to be declared like in C++. It uses parenthesis to indicate an array.

We start with

the maximum number of possible dipole cells	$NMAX$

wavelength	W
number of dipoles in along the major axis	$NSID$
fine structure divisions per dipole cell side	$NLSID$
Particle symmetry semi-axis	RAD
Incident direction	(α, β, γ)
polarization angle	ψ
real and imaginary parts of index of refraction	MR, MI
index of refraction (complex)	M
π	PI
Aspect ratio, ration of major to minor axis	AR

skin depth	SD
dipole course array cell length on a side	TD
number of cell centers on a side	DRF
Fine Array cell size d	LD
Inverse of the fine structure cell volume	$FRAC$
Number of Cells N^3	$NCUB$

number of cells cubed	$NCUB$
number of fine structure cells that are inside the particle	$INCNT$
Number of dipoles that we actually use	$NUSE$
Dipole weighting factors	$D(NMAX)$
Temporary variable, position of fine structure cell	$RF\,(3)$
Locations of dipoles, x, y, and z components.	$R(NMAX, 3)$
$i = \sqrt{-1}$	CI
permittivity	EPS
real and imaginary part of the permittivity	ER, EI

1.5.2 Calculated variables

The skin depth is

$$SD = \frac{W}{2\pi MI}$$

so long as MI is not equal to zero. If not, then $SD = 0$

This comes from assuming an exponential fall off for the plane wave as it enters the dielectric material of the particle

$$E\,(x) = E_o e^{-\alpha x}$$

the skin depth is the distance equal to the reciprocal of the attenuation coefficient

$$x = \frac{1}{\alpha}$$

so that

$$E\,(x) = E_o e^{-1}$$

at this depth. For us

$$\alpha = kn_I = \frac{2\pi}{\lambda} n_I = \frac{2\pi}{W} M_I$$

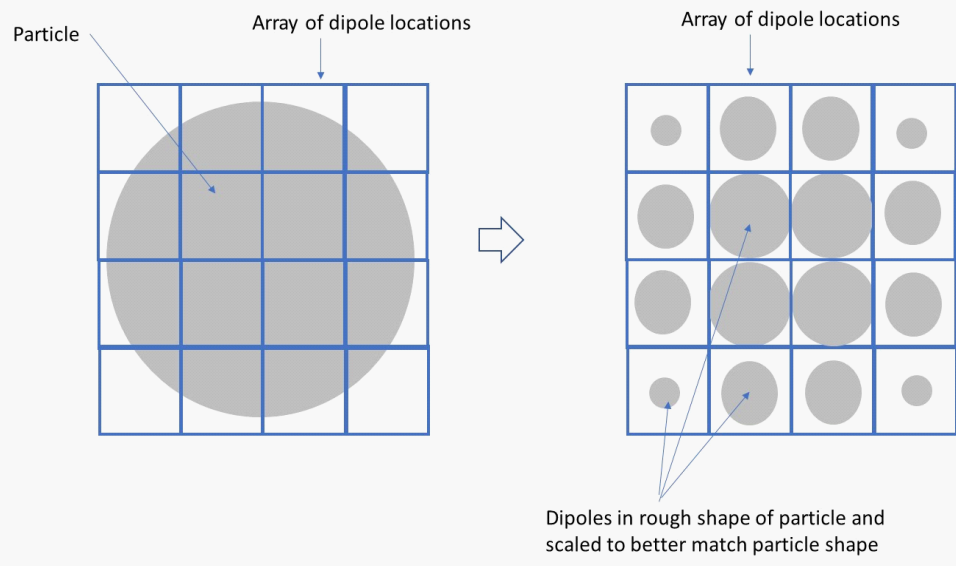
so the skin depth is

$$SD = \frac{W}{2\pi MI}$$

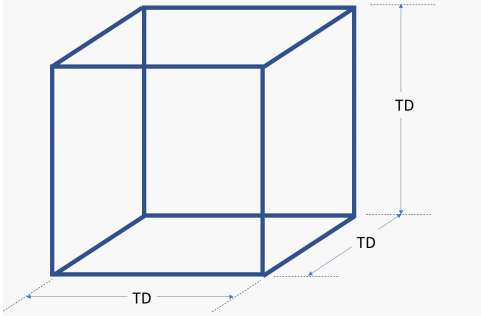
This is interesting, but I don't think it is actually used.

1.5.3 Dipole course array cell length

The code is supposed to make a three dimensional array of dipoles in the shape of the particle. We will split the volume of space into which we put our particle into cubical dipole cells, then figure out which of the dipole cells are actually part of the particle.



The dipole course array cell length on a side is how big the individual dipole cells are.



They are cubical, so the cell volume is defined by the cell length.

$$TD = \frac{2.0 \max\left(RAD, \frac{RAD}{AR}\right)}{NSID}$$

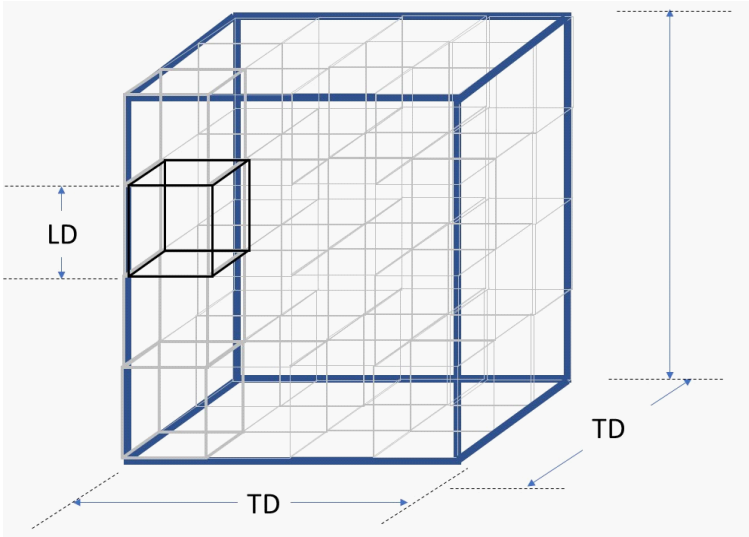
We are going to need to find the center of each of these cells. We will calculate this next bit over and over, so to save some run time, let's precalculate it. More on what it means later.

$$DRF = \frac{1}{2}\left(NSID - 1\right)$$

Now we are going to break up each dipole cell into smaller cells to weight the edge cells by how much of the dipole cell is in the actual particle. To do this we make smaller little cubes inside the dipole cell that have side lengths of d

$$d = \frac{TD}{NLSID} = LD$$

We will call this the "fine structure"



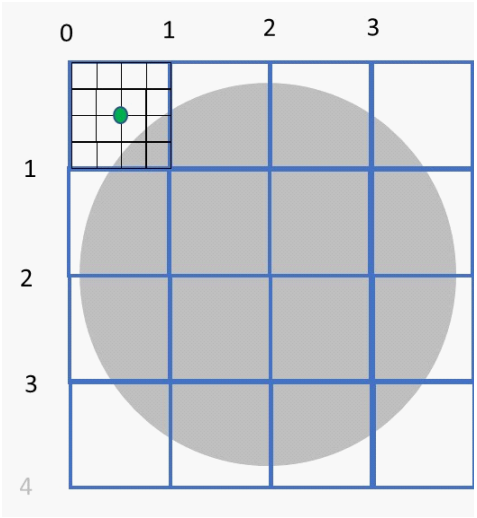
1.5.4 Inverse of the fine structure cell volume

So I precalculated this.

$$FRAC = \frac{1}{NLSID^3}$$

because we will use it later.

1.5.5 Loop to crate particle



The loop is supposed to iterate over each dimension making dipole cell locations. Each dipole cell center location is calculated as

$$x_i = (i - DRF)\, TD$$

Lets try this,. Suppose

$$NSID = 4$$

then

$$\begin{aligned} DRF &= \frac{1}{2}(NSID - 1) \\ &= \frac{1}{2}(4 - 1) \\ &= \frac{3}{2} \end{aligned}$$

then the cell locations in the x direction would be, calculated like this

$$x_i = (i - DRF)\, TD$$

and the first one would be

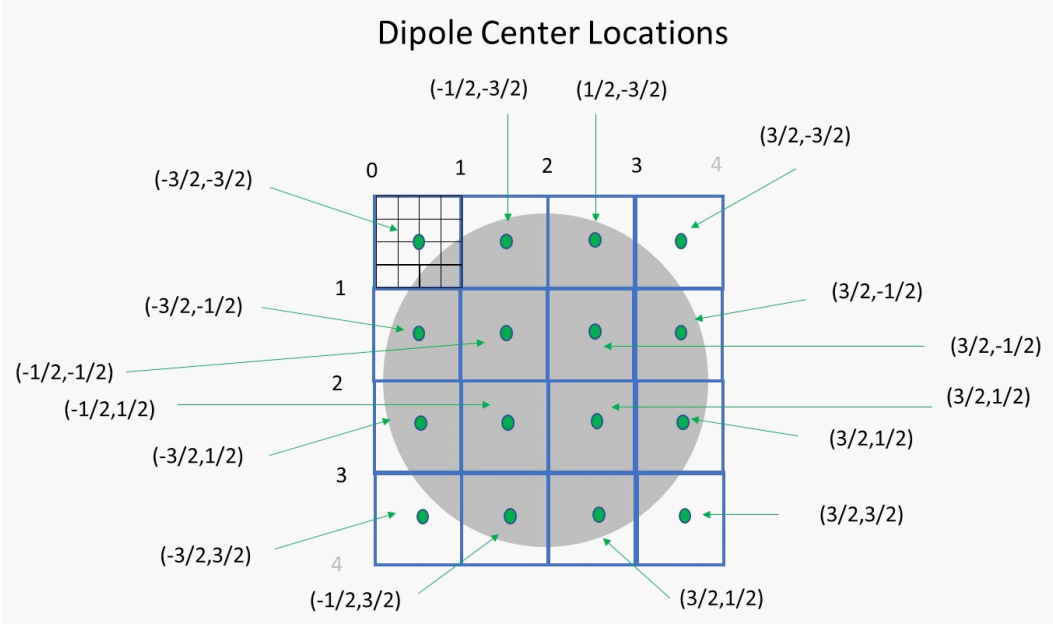
$$x_0 = \left(0 - \frac{3}{2}\right) TD = -\frac{3}{2} TD$$

and here are the rest

i	$x\,(TD)$
0	$-\frac{3}{2}$
1	$-\frac{1}{2}$
2	$\frac{1}{2}$
3	$\frac{3}{2}$

We do this for the y and z directions too.

For the $NSID = 4$ case these would be the dipole cell center locations in the x and y directions (green dots in the next figure)



1.5.6 Fine Structure

Now in the same loop we are going to look at the fine structure boxes to see how many of them are filled with the particle. The fine structure box locations are given by

$$RF(1) = x + LD * (0.5 * (1.0 - NLSID) + IX)$$

for the x direction and similarly for the y and z directions. The quantity IX is the fine structure loop counter. And what we do is to work from the middle of the cube again. Let's take NLSID also equal to 4

$$NLSID = 4$$

then

$$d = \frac{TD}{NLSID} = LD$$

(LD stands for "little d") In our case,

$$d = \frac{TD}{4} = \frac{1}{4}TD$$

gives the side length of the fine structure box. So for our choices

$$RF(1) = x + LD * (0.5 * (1.0 - NLSID) + IX)$$

becomes

$$RF(1) = x + \frac{1}{4}TD * (0.5 * (1.0 - 4) + IX)$$

then if we start with $IX = 0$, we have

$$RF(1) = x + \frac{1}{4}TD * (0.5 * (1.0 - 4) + 0)$$

which would be

$$RF(1) = x + \frac{1}{4}TD * (-\frac{3}{2}))$$

For $IX = 1$

$$RF(1) = x + \frac{1}{4}TD * (0.5 * (1.0 - 4) + 1)$$

$$RF(1) = x + \frac{1}{4}TD * (0.5 * (-3) + 1)$$

$$RF(1) = x + \frac{1}{4}TD * ((-\frac{3}{2}) + 1)$$

$$RF(1) = x + \frac{1}{4}TD * ((-\frac{1}{2})$$

and if $IX = 2$

$$RF(1) = x + \frac{1}{4}TD * (0.5 * (1.0 - 4) + 2)$$

$$RF(1) = x + \frac{1}{4}TD * ((-\frac{3}{2}) + 2)$$

$$RF(1) = x + \frac{1}{4}TD * ((\frac{1}{2})$$

and if $IX = 3$

$$RF(1) = x + \frac{1}{4}TD * (0.5 * (1.0 - 4) + 3)$$

$$RF(1) = x + \frac{1}{4}TD * ((\frac{3}{2})$$

and now let's think about the x values.

i	$x\,(TD)$
0	$-\frac{3}{2}$
1	$-\frac{1}{2}$
2	$\frac{1}{2}$
3	$\frac{3}{2}$

Taking the first for $i = 0$

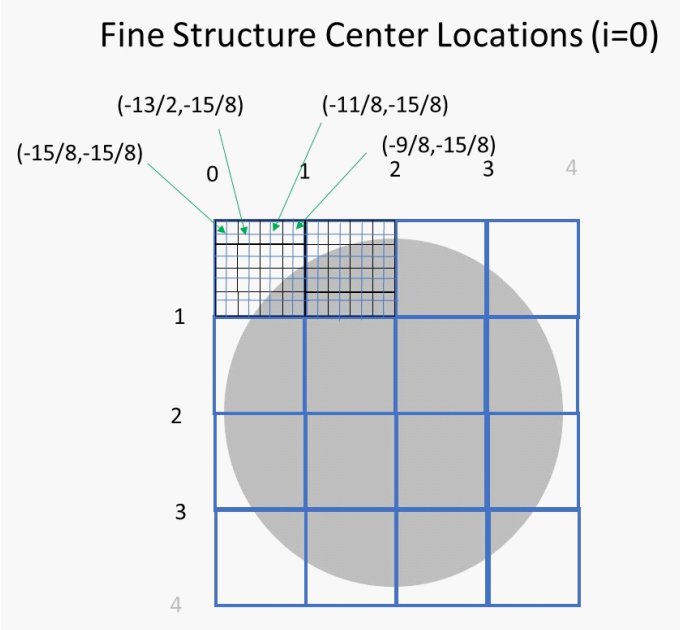
$$\begin{aligned} RF(1) &= -\frac{3}{2}TD + \frac{1}{4}TD * (-\frac{3}{2}) \\ &= -\frac{3}{2}TD + -\frac{3}{8}TD \\ &= -\frac{15}{8}TD \end{aligned}$$

$$RF(1) = -\frac{3}{2}TD + \frac{1}{4}TD * (-\frac{1}{2}) = -\frac{13}{8}TD$$

$$RF(1) = -\frac{3}{2}TD + \frac{1}{4}TD * (\frac{1}{2}) = -\frac{11}{8}TD$$

$$RF(1) = -\frac{3}{2}TD + \frac{1}{4}TD * (\frac{3}{2}) = -\frac{9}{8}TD$$

so the fine structure cell locations would look like this in an end view of our particle space.



and for $i = 1$

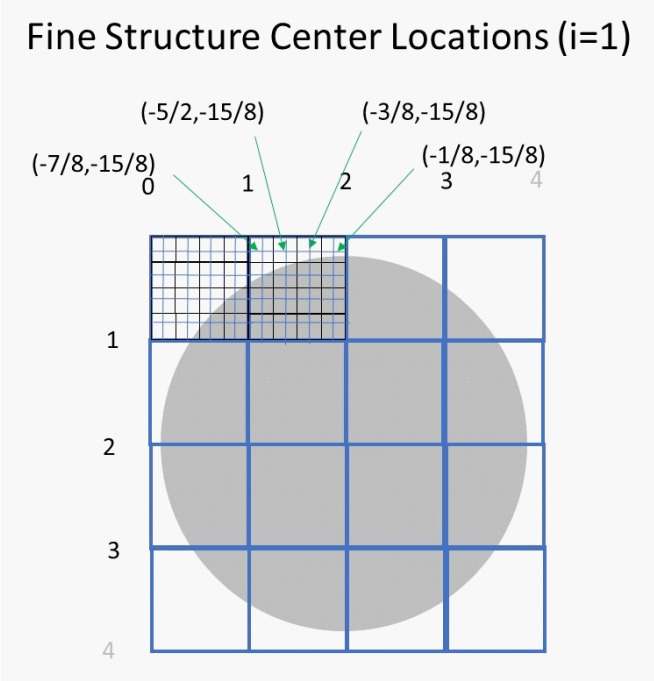
$$RF(1) = -\frac{1}{2}TD + \frac{1}{4}TD * (-\frac{3}{2}) = -\frac{7}{8}TD$$

$$RF(1) = -\frac{1}{2}TD + \frac{1}{4}TD * (-\frac{1}{2}) = -\frac{5}{8}TD$$

$$RF(1) = -\frac{1}{2}TD + \frac{1}{4}TD * (\frac{1}{2}) = -\frac{3}{8}TD$$

$$RF(1) = -\frac{1}{2}TD + \frac{1}{4}TD * (\frac{3}{2}) = -\frac{1}{8}TD$$

which would look like this



Note that we are going to reuse $RF(1)$, $RF(2)$, and $RF(3)$ so the code doesn't put them into an array for every box. We just need to calculate them to see if that location is inside our particle. Then we resuse the RF variable for the next box.

1.5.7 See if a fine structure cell is inside the particle

The whole point of the fine structure is to see how much of the dipole cell is inside the particle. We need to adjust for partially filled dipole cells. Otherwise, all particles would be box like shapes. To do this adjusting we use the function ISINSIDE. This function takes the spheroidal equation

$$1 = \sqrt{\frac{x^2}{b^2} + \frac{y^2}{b^2} + \frac{z^2}{a^2}}$$

where

$$b = \frac{a}{a_r}$$

which defines the outer surface of our ellipsoidal particle. So if

$$1 > \sqrt{\frac{x^2}{b^2} + \frac{y^2}{b^2} + \frac{z^2}{a^2}}$$

we are inside the particle and if

$$1 < \sqrt{\frac{x^2}{b^2} + \frac{y^2}{b^2} + \frac{z^2}{a^2}}$$

we are out side the particle. In the code our ellipsoidal equation is given as

$$1 > \sqrt{\frac{RF\,(1)}{\frac{RAD^2}{AR^2}} + \sqrt{\frac{RF\,(2)}{\frac{RAD^2}{AR^2}} + \sqrt{\frac{RF\,(2)}{RAD^2} +}}}$$

where

$$RAD = a$$

the semimajor axis and

$$\frac{RAD}{AR} = b$$

so that

$$b = \frac{a}{a_r} = \frac{RAD}{AR}$$

If the fine structure location is inside the particle we count it with the counter INCNT. And we do this for all the fine structure locations for the dipole cell. Then we weight the dipole by the number of fine structure cells that are inside the particle.

1.5.8 Dipole weighting

The strategy used to do this is to take the number of fine structure cells that are inside, and weight the size of the dipole based on how many parts of the dipole cell are filled. The scaling factor is given by

$$D = TD\left(\frac{INCNT}{NLSID^3}\right)^{\frac{1}{3}}$$

That is, take the number of fine structure cells that are in the particle and divide by the total number of fine structure cells, and take the cubed root. We have a relative volume, and have converted it into a relative length. Multiply this by TD, the dipole cell size, to get an absolute scaled size for the dipole.

1.6 Print to a file

The program then prints everything to a file. The program uses old fashioned ForTran formatted output. C++ has formatted output. The print statements specify the exact number of digits in each output value. Line 100 is the format. Other write statements are more loose, letting the compiler use the default number of digits.