BAFFLES FROM EXTERNAL PARAMETERS

RICH HOLMES

1. Introduction

I've been familiarizing myself with GEMC 2's provisions for using external parameters to define geometry. As a case study I have created a parameter-driven version of the More1 baffles Perl script. Following are a description of this baffle definition and some observations on its use. Issues regarding incorporation of such parameter-driven geometry into an overall software framework are discussed.

2. The script and its usage

I based my approach on that used in Zhiwen's EC script as discussed at the 26 Mar 2015 software meeting.[1] There is a script called solid_PVDIS_baffle_babarbafflemore1_geometry.pl which sets up the baffle geometry, and a script called solid_PVDIS_baffle.pl that calls its main function, solid_PVDIS_baffle_babarbafflemore1_geometry, whose single argument is a string specifying which variation to use. The latter script is invoked with a command line argument specifying a configuration file, and it reads a parameter file containing names and values for the external parameters controlling the geometry.

Relevant parts of solid_PVDIS_baffle_babarbafflemore1_geometry.pl are shown in Appendix A and solid_PVDIS_baffle.pl in Appendix B. A configuration file is in Appendix C and the parameter file is in Appendix D.

A single Perl script can be used to generate several different versions, or "variations", of the same apparatus. The configuration file specifies a variation, in this case "Original", and the script then reads in parameters from a file tagged with "Original" in the file name. However, in this case the script actually generates several variations by calling the geometry function several times with different arguments. All these variations are based on the "Original" parameters which are available to the script as a hash named %configuration.

The parameters given in the parameter file are the 51 parameters defined in my previous note.[2]. Note that parameters must be scalars. Any sort of array or other structure has to be decomposed into scalar values, e.g. rinin1, rinin3, rinin5... rather than an rinin array.

The main geometry routine uses the parameters to generate the 950 numeric values that define the More1 baffle geometry. In addition, it sets the materials to be used and the sensitivity based on the variation name: If the name contains "Kill" the materials are Kryptonite and Vacuum, and the baffles are not sensitive; if it contains "Sens" the material is Vacuum and the sensitivity is flux; otherwise the materials are G4_Pb and G4_Air and the baffles are not sensitive.

After setting up the geometry values the main routine calls routines to generate the inner ring, outer ring, plates, and blocks within the plates which define the slits. If the variation name contains "NoInner" then the inner ring is omitted.

Not seen in Appendix A is a section of code which, if the variation name contains "Block", adds material to the last baffle to act as a photon blocker.

These various alternatives are orthogonal, so for example one could request a variation called "NoInnerKillBlock" with no inner ring, Kryptonite baffles, and a photon blocker.

The outputs of the Perl scripts are text geometry files with names like "solid_PVDIS_baffle__geometry_NoInner.txt" each containing the detailed, GEANT4-based description of one baffle variation ("NoInner" in this case). In a gcard file one can include baffles in the geometry using a line like

```
<detector name="solid_PVDIS_baffle" factory="TEXT"
variation="NoInnerSens"/>
```

This tells GEMC to get the "NoInnerSens" variation of the baffle from the text file.

3. Some observations

I have found the external parameter-driven baffles to be a great improvement over what I had been doing in GEMC 1.x. Previously in order to use any variant of the standard baffles I had to either modify the baffle script, run it, and upload it to my user_geometry MySQL database (an obviously risky practice if one loses track of which variant is currently stored) or copy the script and edit it, assigning a new name to the detector, run that, and upload that as a new database table; this results in a proliferation of scripts and database tables, difficult to maintain. With the parameter-driven baffles one still ends up with multiple text files, but only a single script to be maintained. As a separate issue, using the TEXT factory method rather than having to deal with a remote database is a great improvement.

There still are some of what I would consider design flaws in this approach.

As I wrote the script, the variations consist of hardwired (in the Perl script) modifications to the "Original" geometry; they are not described by separate parameter files. They could be: the materials, sensitivity, and photon blocker could be specified by parameters. The script would have no conditionals based on the variation name but simply would read the appropriate parameter file and use it to generate the geometry. However, this would then create maintenance problems since there would be multiple parameter files. Each new variation would require manual creation of a new file to be run through the script. Working with a single parameter file and incorporating variations via script conditionals, as I have done, makes creation and maintenance of multiple variations easier; the drawback is that there is no concise specification of the variations. That is, to understand what the geometry is you need to understand the script's conditionals. Presumably if the simulation output is to contain a specification of the geometry, it would need either the detailed geometry file or the parameter file plus the variation name plus something identifying the Perl script version in the software repository.

This setup is problematic in the context of a full simulation / digitization / calibration / reconstruction framework. The output of the script is a detailed baffle description in

GEANT-centric style which is not appropriate for digitization, calibration, or reconstruction. Those pieces of the framework would need different descriptions. Using the present parameter file, there are three obvious possibilities. The C++ baffle class used by digitization, calibration, and reconstruction could read the parameter file and generate appropriate descriptions internally; other Perl scripts could read the file and generate alternate descriptions as text output files, to be read by the C++ code; or the GEMC Perl script could be augmented to generate multiple descriptions in multiple text files. All these approaches have obvious shortcomings. For example, if the C++ baffle class or a separate Perl script must generate the baffle descriptions from the parameter file, then that class or script must be maintained in synch with the GEMC Perl script; in particular, any variations generated by conditionals in the GEMC script must be replicated in the C++ or other Perl script. This would potentially be an implementation and maintenance nightmare.

Having a single script generate all geometry descriptions encapsulates all knowledge of how to get from parameter file and variation name to detailed geometry in one place. However, in either this or the multiple Perl scripts scenario, one then needs to define and maintain text file formats for each geometry description, Perl code to write those formats, and C++ code to read them.

A fourth alternative is to abandon Perl scripts and instead have parameter files directly read by a C++ baffle class which is used by all parts of the framework including simulation. There again knowledge of how to go from parameters to detailed geometry is encapsulated, but without having to go through intermediate text files. A ROOT streamer can be used to write geometry descriptions to files if needed. This however would require significant modifications to GEMC if we choose to continue using GEMC in our framework.

References

[1] Zhiwen Zhao, "SoLID simulation with GEMC" (26 Mar 2015)
http://hallaweb.jlab.org/12GeV/SoLID/download/software/talk/solid_software_zwzhao_20150326.pdf.
[2] Richard Holmes, "Parameterization of More1 baffles" (17 Jun 2015)
https://hallaweb.jlab.org/wiki/images/e/e8/Baffle_parameterization.pdf.

Appendix A. solid_PVDIS_baffle_babarbafflemore1_geometry.pl

This abbreviated listing shows just global variables and the main routine.

```
#!/usr/bin/perl -w
use strict;
use warnings;
our %detector;
our %configuration;
our %parameters;

use Getopt::Long;
use Math::Trig;
```

```
my $DetectorName = 'solid_PVDIS_baffle_babarbafflemore1p';

my $DetectorMother="root";

# Global variables governing some of the variations
my $color_baffle;
my $sensitivity_baffle;
my $hit_baffle;
my $material_baffle;
my $material_baffle_within;

# System parameters
my $Nplate; ## of baffles
my $Dz;  # thickness
my $Nslit;  # of slits
my $Nblock; ## of radial layers
my $zc0; # center of baffle system
my $Dzc;  # baffle spacing

# Baffle parameters
my @rinin;  # inner radius of inner ring
my @rinout; # outer radius of inner ring
my @routin;  # inner radius of outer ring
my @routout; # outer radius of outer ring
my @offset;  # phi offset

# c0, c1, c2 are constant, linear, quadratic coefficients
# Slit parameters
my $c0_rint;   # intersection radius
my $c1_rint;
my $c0_phiint;   # starting phi at rint
my $c1_phiint;
my $c0_Dphiint;   # angular width at rint
my $c1_Dphiint;
my $c0_im0phi;   # inverse of starting phi slope below rint
my $c1_im0phi;
my $c0_im1phi;   # inverse of starting phi slope above rint
my $c1_im1phi;
my $c0_im0Dphi;   # inverse of angular width slope below rint
my $c1_im0Dphi;
```

```
my $c2_im0Dphi;
my $c0_im1Dphi;   # inverse of angular width slope above rint
my $c1_im1Dphi;

my @PlateZ;

sub solid_PVDIS_baffle_babarbafflemore1_geometry
{
# the first argument to this function becomes the variation
    $configuration{"variation"} = shift;

    if ($configuration{"variation"} =~ /Kill/)
    {
        # Inert Kryptonite
        $material_baffle = "Kryptonite";
        $material_baffle_within = "Vacuum";
        $sensitivity_baffle="no";
        $hit_baffle="no";
    }
    elsif ($configuration{"variation"} =~ /Sens/)
    {
        # Sensitive vacuum
        $material_baffle = "Vacuum";
        $material_baffle_within = "Vacuum";
        $sensitivity_baffle="flux";
        $hit_baffle="flux";
    }
    else
    {
        # Inert lead
        $material_baffle = "G4_Pb";
        $material_baffle_within = "G4_AIR";
        $sensitivity_baffle="no";
        $hit_baffle="no";
    }

    $color_baffle="00C0C0";
    $Nplate  = $parameters{"Nplate"};
    $Dz   = $parameters{"Dz"};
    $Nslit  = $parameters{"Nslit"};
```

```
    $Nblock  = $parameters{"Nblock"};
    $zc0 = $parameters{"zc0"};
    $Dzc = $parameters{"Dzc"};
    @rinin = ($parameters{"rinin1"}, 0.0, $parameters{"rinin3"}, 0.0,
              $parameters{"rinin5"}, 0.0, $parameters{"rinin7"}, 0.0,
              $parameters{"rinin9"}, 0.0, $parameters{"rinin11"});
    @rinout = ($parameters{"rinout1"}, 0.0, $parameters{"rinout3"}, 0.0,
               $parameters{"rinout5"}, 0.0, $parameters{"rinout7"}, 0.0,
               $parameters{"rinout9"}, 0.0, $parameters{"rinout11"});
    @routin = ($parameters{"routin1"}, 0.0, $parameters{"routin3"}, 0.0,
               $parameters{"routin5"}, 0.0, $parameters{"routin7"}, 0.0,
               $parameters{"routin9"}, 0.0, $parameters{"routin11"});
    @routout = ($parameters{"routout1"}, 0.0, $parameters{"routout3"}, 0.0,
                $parameters{"routout5"}, 0.0, $parameters{"routout7"}, 0.0,
                $parameters{"routout9"}, 0.0, $parameters{"routout11"});
    @offset = ($parameters{"offset1"}, 0.0, $parameters{"offset3"}, 0.0,
               $parameters{"offset5"}, 0.0, $parameters{"offset7"}, 0.0,
               $parameters{"offset9"}, 0.0, $parameters{"offset11"});

# Even baffles by interpolation
    for (my $ip = 1; $ip < $Nplate; $ip += 2)
    {
        $rinin[$ip] = 0.5 * ($rinin[$ip-1] + $rinin[$ip+1]);
        $rinout[$ip] = 0.5 * ($rinout[$ip-1] + $rinout[$ip+1]);
        $routin[$ip] = 0.5 * ($routin[$ip-1] + $routin[$ip+1]);
        $routout[$ip] = 0.5 * ($routout[$ip-1] + $routout[$ip+1]);
        $offset[$ip] = 0.5 * ($offset[$ip-1] + $offset[$ip+1]);
    }

    $c0_rint = $parameters{"c0_rint"};
    $c1_rint = $parameters{"c1_rint"};
    $c0_phiint = $parameters{"c0_phiint"};
    $c1_phiint = $parameters{"c1_phiint"};
    $c0_Dphiint = $parameters{"c0_Dphiint"};
    $c1_Dphiint = $parameters{"c1_Dphiint"};
    $c0_im0phi = $parameters{"c0_im0phi"};
    $c1_im0phi = $parameters{"c1_im0phi"};
    $c0_im1phi = $parameters{"c0_im1phi"};
    $c1_im1phi = $parameters{"c1_im1phi"};
    $c0_im0Dphi = $parameters{"c0_im0Dphi"};
```

```
    $c1_im0Dphi = $parameters{"c1_im0Dphi"};
    $c2_im0Dphi = $parameters{"c2_im0Dphi"};
    $c0_im1Dphi = $parameters{"c0_im1Dphi"};
    $c1_im1Dphi = $parameters{"c1_im1Dphi"};

    @PlateZ = ();
    for (my $i = 0; $i < $Nplate; ++$i)
    {
        my $z = $zc0 + $Dzc * ($i - ($Nplate-1) * 0.5);
        push @PlateZ, $z;
    }

    if ($configuration{"variation"} !~ /NoInner/)
    {
        make_baffle_plate_inner();
    }
    make_baffle_plate_outer();
    make_baffle_plate();
    make_baffle_blocks();
}
```

Appendix B. solid_PVDIS_baffle.pl

```
#!/usr/bin/perl -w

use strict;
use lib ("$ENV{GEMC}/io");
use parameters;
use utils;

use geometry;
use hit;
use bank;
use math;

use Math::Trig;
# use Math::MatrixReal;
# use Math::VectorReal;

# system("rm meic_det1_simple__*txt");
```

```
# Help Message
sub help()
{
        print "\n Usage: \n";
        print "   detector.pl <configuration filename>\n";
        print "   Will create the detector\n";
        print "   Note: The passport and .visa files must be present to conne
        exit;
}

# Make sure the argument list is correct
if( scalar @ARGV != 1)
{
        help();
        exit;
}


# Loading configuration file and parameters
my $config_file   = $ARGV[0];
our %configuration = load_configuration($config_file);

# One can change the "variation" here if one is desired different from the co
$configuration{"detector_name"} = "solid_PVDIS_baffle";
$configuration{"variation"} = "Original";

# To get the parameters proper authentication is needed.
our %parameters    = get_parameters(%configuration);

#Geometry definition
require "solid_PVDIS_baffle_babarbafflemore1_geometry.pl";
solid_PVDIS_baffle_babarbafflemore1_geometry ("Original");
solid_PVDIS_baffle_babarbafflemore1_geometry ("Kill");
solid_PVDIS_baffle_babarbafflemore1_geometry ("NoInner");
solid_PVDIS_baffle_babarbafflemore1_geometry ("NoInnerKill");
solid_PVDIS_baffle_babarbafflemore1_geometry ("NoInnerKillBlock");
solid_PVDIS_baffle_babarbafflemore1_geometry ("NoInnerSens");

#hit and bank definition Execute only when there are changes
```

```
#hit
#require "./solid_baffle_hit.pl";
#define_solid_baffle_hit();

# banks
#require "./solid_baffle_bank.pl";
#define_solid_baffle_bank();
```

Appendix C. config_solid_baffle.dat

```
# Configuration file for new_detector

# Detector name and variation
detector_name: solid_PVDIS_baffle
variation:     Original

# Factory can be MYQL or TEXT
factory:       TEXT

# run ranges and variation will apply to geometry, materials and parameters
rmin:       1
rmax:       10000

# MYSQL server and databse if MYSQL factory is selected
dbhost: localhost

# Comments/Description
comment:       solid_PVDIS_baffle

# Verbosity controls the perl script output
verbosity: 1
```

Appendix D. solid_PVDIS_baffle__parameters_Original.txt

Nplate	11		Nplate	-	-	-	-	-	-
Dz	4.5	cm	Dz	-	-	-	-	-	-
Nslit	30		Nslit	-	-	-	-	-	-
Nblock	20		Nblock	-	-	-	-	-	-
zc0	110.0	cm	zc0	-	-	-	-	-	-
Dzc	14.0	cm	Dzc	-	-	-	-	-	-
rinin1	3.89	cm	rinin1	-	-	-	-	-	-
rinin3	14	cm	rinin3	-	-	-	-	-	-

rinin5	24	cm	rinin5	–	–	–	–	–
rinin7	35.9	cm	rinin7	–	–	–	–	–
rinin9	47.9	cm	rinin9	–	–	–	–	–
rinin11	58.8	cm	rinin11	–	–	–	–	–
rinout1	3.9	cm	rinout1	–	–	–	–	–
rinout3	15.3	cm	rinout3	–	–	–	–	–
rinout5	26.6	cm	rinout5	–	–	–	–	–
rinout7	37.9	cm	rinout7	–	–	–	–	–
rinout9	49.2	cm	rinout9	–	–	–	–	–
rinout11	60.4	cm	rinout11	–	–	–	–	–
routin1	34.7	cm	routin1	–	–	–	–	–
routin3	54.3	cm	routin3	–	–	–	–	–
routin5	73.8	cm	routin5	–	–	–	–	–
routin7	93.3	cm	routin7	–	–	–	–	–
routin9	112.8	cm	routin9	–	–	–	–	–
routin11	132	cm	routin11	–	–	–	–	–
routout1	42	cm	routout1	–	–	–	–	–
routout3	62	cm	routout3	–	–	–	–	–
routout5	82	cm	routout5	–	–	–	–	–
routout7	102	cm	routout7	–	–	–	–	–
routout9	120	cm	routout9	–	–	–	–	–
routout11	140	cm	routout11	–	–	–	–	–
offset1	−5.6	deg	offset1	–	–	–	–	–
offset3	−4.4	deg	offset3	–	–	–	–	–
offset5	−3.3	deg	offset5	–	–	–	–	–
offset7	−2.1	deg	offset7	–	–	–	–	–
offset9	−0.9	deg	offset9	–	–	–	–	–
offset11	0.1	deg	offset11	–	–	–	–	–
c0_rint	55.83	cm	c0_rint	–	–	–	–	–
c1_rint	0.6967		c1_rint	–	–	–	–	–
c0_phiint	178.1	deg	c0_phiint	–	–	–	–	–
c1_phiint	0.02252	deg cm^−1	c1_phiint	–	–	–	–	–
c0_Dphiint	4.19	deg	c0_Dphiint	–	–	–	–	–
c1_Dphiint	−0.01257	deg cm^−1	c1_Dphiint	–	–	–	–	–
c0_im0phi	8.372	cm deg^−1	c0_im0phi	–	–	–	–	–
c1_im0phi	0.001967	deg^−1	c1_im0phi	–	–	–	–	–
c0_im1phi	10.53	cm deg^−1	c0_im1phi	–	–	–	–	–
c1_im1phi	−0.006294	deg^−1	c1_im1phi	–	–	–	–	–
c0_im0Dphi	−8.868	cm deg^−1	c0_im0Dphi	–	–	–	–	–
c1_im0Dphi	−0.08631	deg^−1	c1_im0Dphi	–	–	–	–	–

c2_im0Dphi	0.0002641	cm^-1 deg^-1	c2_im0Dphi	-	-	-	-	-
c0_im1Dphi	-54.56	cm deg^-1	c0_im1Dphi	-	-	-	-	-
c1_im1Dphi	0.03047	deg^-1	c1_im1Dphi	-	-	-	-	-