The Description of `LArWheelSolid` Version 0.1.0

Andrei Soukharev

5 April 2001

1 Introduction

This paper describes a new Geant4[1] solid, simulating the geometry of absorbers and electrodes of the electromagnetic end-cap calorimeter[2] of the ATLAS detector.

2 The geometry of the LAr EM end-cap calorimeter

Below is the description of the calorimeter geometry. This description represents my understanding of calorimeter structure, and the implementation of `LArWheelSolid` is based on it. Only the most complex parts of the calorimeter are involved in the implementation: the absorbers and the electrodes. Other parts of the end-cap calorimeter have yet to be described in the Geant4 detector geometry routines.

In addition to papers mentioned below, I've also used papers [3, 4, 5] to understand how the calorimeter is organized. The reader is refered to the paper [2] for images of various parts of the calorimeter.

2.1 Theory

A single calorimeter end-cap consists of two concentric wheels: inner and outer. These wheels are composed of absorbers and electrodes positioned radially like spokes. Absorbers and electrodes have a fan-like shape, with the fans stretched such that their sides are parallel (they are connected to the front and rear calorimeter surfaces).

Absorbers and electrodes geometrically differ only by thickness, so I'll use the common term **"fan"** instead of the words "absorber" and "electrode."

The calorimeter has an axial symmetry, so it is sufficient to set up the geometry of a single fan. Fans are positioned with a polar angle step of $2\pi/n_{fan}$, where n_{fan} is the number of fans in a wheel. The end-cap is divided radially into inner and outer wheels. The boundary between the inner and outer wheel,

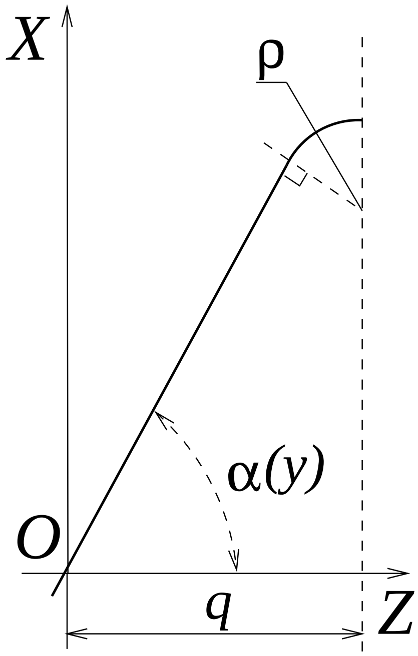

Figure 1: 1/4 of wave of the fan's neutral fibre. O – local origin of coordinates, q – length of 1/4 of fan's wave.

and the inner radius of the inner wheel, are angled to project to the detector center. Thus, while the overall shape of the endcap is a disk, the inner and outer wheels are cone-shaped.

The amplitude of a fan's waves increases with distance from the wheel axis. The wave slant angle α has a complex dependency on the radius.

The description of the geometry is based on the **neutral fibre** of fan, defined as a wave-like three-dimensional surface in the middle of the fan thickness. Fig. 1 represents the cross-section of the neutral fibre in the plane parallel to the plane OZX and displaced from the calorimeter axis at a distance y. 1/4 of a neutral fibre's wave is shown. It consists of a straight region, where $x = z \tan \alpha(y)$, and a fold region. The fold radius ρ doesn't depend on the distance to the calorimeter axis.

The geometry of the first and last quarter-waves of a fan is a little different (Fig. 2). The fold region that directly connects to the front or rear wheel surface has radius ρ_1 that depends on the distance to the calorimeter axis. The radius is expressed in terms of the wave slant angle α in the following way: $\rho_1 = s \tan \beta, \beta = (\pi - \alpha(y))/2$, where s is the distance from a fold middle to a fan boundary. So the first and last quarter-waves are s longer.

All geometrical calculations in `LArWheelSolid` are based on the computation of the distance to the neutral fibre with the properties described above.

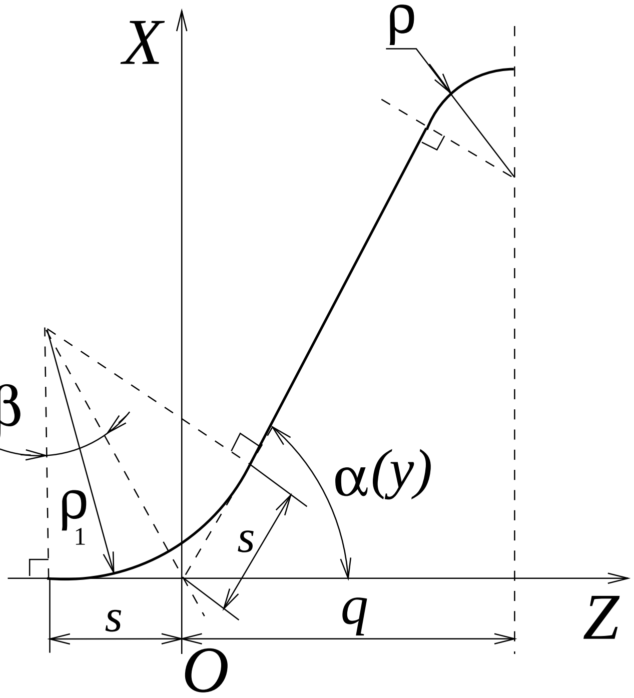

Figure 2: 1/4 of the first wave of the fan's neutral fibre. O – local origin of coordinates, q – length of 1/4 fan's wave.

2.2 Practice

A practical realization of a model of the geometry is based on the paper [6], which contains tables of neutral fibre sizes depending on the distance to the calorimeter axis, and the program [7], where the fan wave slant angle α dependence on the distance to the calorimeter axis is parameterized with a 4th-degree polynomial.

Real absorbers and electrodes have a structure of several layers, with each layer made of different materials. In `LArWheelSolid` the multilayer structure is replaced with a single-layer one using a "mean" material; this is similar to the approach used in `LArBarrelSolid`[8].

Some geometrical constants included in the code are given in Table 1.

Table 1: Some geometrical parameters of fan's neutral fibre.

	inner wheel	outer wheel
number of waves in a fan	6	9
wheel thickness	514 mm	
starting and finishing fan fold regions s	2 mm	
fold radius ρ	3.25 mm	3.0 mm
nomber of fans n_{fan} in a wheel	256	768
polar angle of a fan	2×0.055	2×0.015
absorber thickness (lead + steel + glue)	$(2.2 + 0.4 + 0.3)$ mm	$(1.7 + 0.4 + 0.3)$ mm
full thickness of an electrode	275 μm	

parameterization of the angle $\alpha(y) = a_0 + a_1 y + a_2 y^2 + a_3 y^3 + a_4 y^4$

	a_0	a_1	$a_2 \times 10^3$	$a_3 \times 10^7$	$a_4 \times 10^{11}$
outer wheel	-34.254	0.15528	-0.1167	0.45018	-0.68473
inner wheel	-50.069	0.50073	-1.0127	10.39	-42.176

I didn't check if the parametrization constants for the angle α are correct. In addition, the bounding sizes of the wheels used in the program are not exact.

3 Geant4 requirements

In the Geant4 package a solid is a class; when you create a new type of solid it must be inherited from the base class `G4VSolid` and the following methods must be implemented [9]:

- `Inside(`$\vec{p}$`)` — determines if the point p is inside the solid, on its surface, or outside.
- `DistanceToIn(`$\vec{p}$`)`, `DistanceToOut(`$\vec{p}$`)` — determines the shortest distance from the point p located outside or inside the solid (respectively) to the solid surface.

- `SurfaceNormal`($\vec{p}$) — determines a vector normal to the solid surface and going through the point p.
- `DistanceToIn`($\vec{p}, \vec{v}$), `DistanceToOut`($\vec{p}, \vec{v}$) — computes the distance from the point p located outside or inside the solid (respectively) to the solid surface along the arbitrary vector $\vec{v}$.
- `GetEntityType`, `DescribeYourselfTo`, `GetExtent`, `CreatePolyhedron`, `CreateNURBS` — special functions for tracking and solid drawing.

Distances returned by both of types of routines `DistanceToIn` and `DistanceToOut` can be underestimates.

4 Algorithms and data of `LArWheelSolid`

Due to the calorimeter axial symmetry, all the input parameters are translated to the coordinate system where the fan is positioned along the OY axis. All the further calculations are conducted in this coordinate system relative to this "vertical" fan. If it is necessary, after the calculation the results are translated back to the inital coordinate system.

It is recommended that the `LArWheelSolid` code (files `src/LArWheelSolidInit.cc`, `src/LArWheelSolid.cc`, `include/LArWheelSolid.hh`) be consulted while reading this paper.

In Table 2 each important data element is listed with a short comment about its purpose. These names are used in the text below without any additional references.

4.1 `LArWheelType`

`LArWheelType` — special enumerator type. Variables of this type can be one of four values: `InnerAbsorberWheel`, `InnerElectrodWheel`, `OuterAbsorberWheel`, `OuterElectrodWheel`. The constructor takes a parameter of this type; it indicates which type of wheel is being defined: inner or outer, absorber or electrode. It is assumed that physical volumes made of each type of wheel would be positioned at a common place (at the vector $(0, 0, Z_0)$). If so, the relative positions will be correct: electrodes positioned between absorbers of the corresponding wheel, the outer wheel surrounds the inner one.

4.2 `distance_to_neutral_fibre`($\vec{p}$)

This function is used by most of the class's methods. It returns the shortest distance from the point p to the neutral fibre of the vertical fan. The returned value is positive if p is positioned in the region of relatively greater polar angle to the neutral fibre, and negative otherwise.

The function uses an approximation: for the calculation of the angle $\alpha(y)$, the y-coordinate of p is used instead of the correct value of the y-coordinate of

Table 2: Data elements used in `LArWheelSolid`

`LArWheelType Type`	wheel type (see subsection 4.1)
`G4double FanHalfThickness`	half of a fan thickness
`static const G4double WheelThickness`	wheel thickness (along Z axis)
`static const G4double StraightStartSection`	size of starting and ending fan fold regions (s)
`G4double QuarterWaveLength`	quarter of a fan wave length
`G4double HalfWaveLength`	half of a fan wave length
`G4double FanFoldRadius`	fold radius (ρ)
`G4int NumberOfWaves`	number of waves in a fan
`G4int NumberOfFans`	number of fans in the wheel (n_{fan})
`G4double ZeroFanPhi`	polar angle of fan number zero
`G4double FanStepOnPhi`	step in ϕ between fans ($2\pi/n_{fan}$)
`static const G4double Tolerance`	precision of solid boundary determination
`static const G4double IterationPrecision`	precision required to stop iteration calculation
`static const G4double IterationPrecision2`	second power of `IterationPrecision`
`static const unsigned int IterationsLimit`	maximal number of iterations; required for loop breaking in case of an error
`G4Polycone *BoundingPolycone`	see subsection 4.7
`G4Polycone **FanSection`	see subsection 4.12

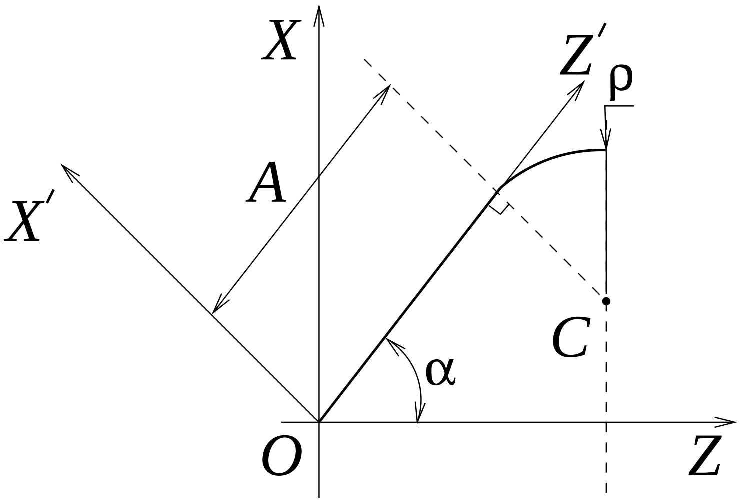

Figure 3: Calculation of distance d from point (z, x) to the neutral fibre. If $z' < A$, then $d = x'$, else $d = \sqrt{(x' - C_{x'})^2 + (z' - C_{z'})^2} - \rho$.

the point on the neutral fibre nearest to p. The calculations are made in the plane parallel to plane OZX and going through the point p. The routine makes use of the symmetry of waves of the neutral fibre about the "knot" (the point of inflection at O in Fig. 1).

Using p_z the function determines the quarter-wave of the neutral fibre nearest to p. If this is neither first nor last quarter-wave, the coordinates (p_z, p_x) are tranformed in the following way:

1. a shift on z is performed, so the local coordinate origin O moves to the knot point,
2. odd half-waves are mirrored relative to the local OZ axis,
3. odd quarter-waves are additionally inverted relative to the local coordinate origin O.

After these transformations, all the initial cases are moved to a standard co-ordinate system: the nearest quarter-wave where local coordinates z and x are positive (Fig. 1).

Next the local coordinate system is rotated, so that the straight part of the quarter-wave is positioned along Z' axis (Fig. 3). In this coordinate system it is easy to determine if the initial point is nearer to the straight part of the quarter-wave or to the fold region. In both cases the distance is computed using simple exact two-dimensional formulae.

At the end the result is given with the correct sign, depending on which transformations were performed.

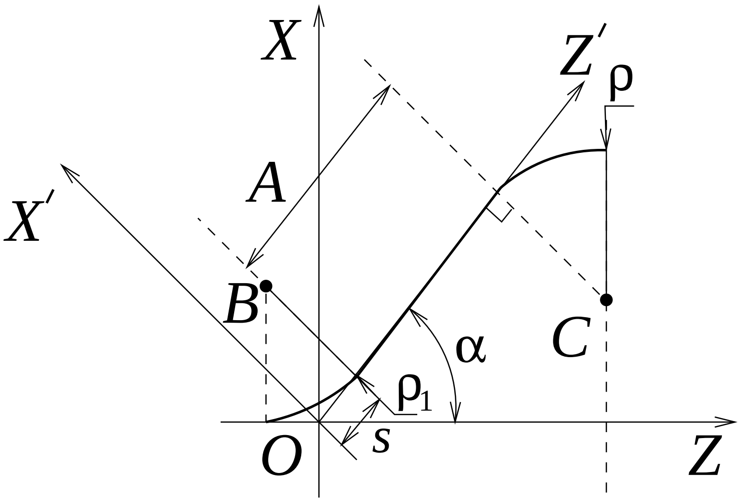

Figure 4: Calculation of the distance d from point (z, x) to the fan's neutral fibre, the case of the starting quarter-wave. If $z' < s$, then $d = \sqrt{(x' - B_{x'})^2 + (z' - B_{z'})^2} - \rho_1$, if $s \le z' < A$, then $d = x'$, else $d = \sqrt{(x' - C_{x'})^2 + (z' - C_{z'})^2} - \rho$.

In the case of first and last quarter-wave the situation becomes more complex due to the existence of the second fold region. Using symmetry the last quarter-wave is transformed to the first one. The next steps are almost the same as the ones for the regular quarter-wave; the only exception is that the local coordinate origin is positioned while transforming not into the beginning of the quarter-wave, but into the cross point of the straight part that continues into the local OZ axis (Fig. 4).

According to measurements, the error of the distance to the neutral fibre determined by this technique is not more than several tens of microns. It is sufficient for most applications.

4.3 `nearest_point_on_neutral_fibre`($\vec{p}$)

This function is nearly identical to the function `distance_to_neutral_fibre` and uses the same approximation technique. The function returns the point on the vertical fan neutral fibre which is the "nearest" to p as defined by `distance_to_neutral_fibre`.

If we need a more precise answer, it is necessary to rewrite this function.

4.4 `inside_vertical_fan`($\vec{p}$), `inside_vertical_fan_exact`($\vec{p}$)

The function `inside_vertical_fan`($\vec{p}$) determines if the point p is inside, on the surface, or outside the vertical fan. The approach is very simple: if the distance to the neutral fibre is less than the `FanHalfThickness` with the precision of the `Tolerance` then the point is inside, if the distance is greater it is outside, otherwise it is on the surface. The `distance_to_neutral_fibre` function is used.

The function `inside_vertical_fan_exact`($\vec{p}$) is the same as `inside_vertical_fan` at present. If the precision of the `distance_to_neutral_fibre` is not be enough in some cases, we should modify the `inside_vertical_fan_exact` to provide the precision required.

4.5 `determine_nearest_fans`($\vec{p}$)

The function determines the distance between a point p and its two nearest fans, and returns the number of the closest fan i. The point p will be positioned in the region of greater polar angle ϕ relative to fan i's neutral fibre. It is guaranteed the point p is located in the region of smaller ϕ relative to the neutral fibre of fan number $i + 1$.

The algorithm is the following. First of all, using the formula

$$i = \mathrm{int}((p_\phi - \pi/2 - \texttt{ZeroFanPhi})/\texttt{FanStepOnPhi})$$

the number i of one of the nearest fans is determined. Next, the neighboring fans are checked one by one until the fan's pair will be found; when the distance from p to the neutral fibre of the first fan of the pair is positive and the distance to the second one is negative, it means that p is situated between the two fans.

4.6 `rotate_vector_to_vertical_fan`($\vec{p}$)

The function transforms the point p into the coordinate system where the vertical fan becomes the fan with the number returned by the `determine_nearest_fans`($\vec{p}$). The result is then rotated relative to the Z axis by $-$`FanStepOnPhi` to guarantee at that the ϕ of point p is always greater than the ϕ of the vertical fan.

4.7 `BoundingPolycone`

`BoundingPolycone` in object of class `G4Polycone`, which represents a wheel surface. The `G4Polycone` defines the boundaries of the `LArWheelSolid`. There is no need to take care of fan boundaries in the Z and radial directions, because for points outside the `BoundingPolycone` none of the routines in `LArWheelSolid` would be called.

The `BoundingPolycone` is also used in special functions of the Geant4 tracking and solid drawing routines. To any such routine, `LArWheelSolid` looks like `BoundingPolycone`.

`BoundingPolycone` is initialized in the function `inner_wheel_init` or `outer_wheel_init`.

4.8 `Inside`$(\vec{p})$

The function checks if the point p is inside the `BoundingPolycone`. If it is, then it determines the two fans nearest to p and checks if the point is inside or on the surface of one them.

4.9 `DistanceToIn`$(\vec{p})$

The function checks if the point p is inside the `BoundingPolycone`. If yes, then it determines the two fans nearest to p, calculates the distances to each of them, and returns the smallest. Otherwise it passes p to the `DistanceToIn`$(\vec{p})$ method of `BoundingPolycone`.

4.10 `DistanceToOut`$(\vec{p})$

The function checks if the point p is inside the `BoundingPolycone`. If yes, then it determines in which of the two fans nearest to p the point is located and returns the distance to its surface. If the point is not inside any of the fans or it is outside the `BoundingPolycone` the function returns zero.

4.11 `SurfaceNormal`$(\vec{p})$

The function checks if the point p is inside the `BoundingPolycone`. If yes, then it determines the nearest fan, finds the nearest point on the fan's neutral fibre, and returns the unit vector pointing to p from this point. Otherwise it passes p to the `SurfaceNormal`$(\vec{p})$ method of `BoundingPolycone`.

4.12 `FanSection`

The array `FanSection` contains objects of type `G4Polycone` each of which corresponds to the vertical fan's half-wave of the current wheel. `FanSection[i]` is a polyconical sector containing the whole i-th half-wave of the vertical fan. Its Z-boundaries are the same as that of the the vertical fan; its ϕ-boundaries are enough to capture all the points between the vertical fan and its nearest neighbors.

The `FanSection[i]` objects are used by the functions that calculate the distance to the solid surface along an arbitrary vector.

The initialization of the `FanSection` array is made by function `inner_wheel_init` or `outer_wheel_init`.

4.13 `out_iteration_process`$(\vec{P}, \vec{Q})$

The function searches for the exit point of segment PQ from the fan's half-wave in the event that Q is not inside the fan.

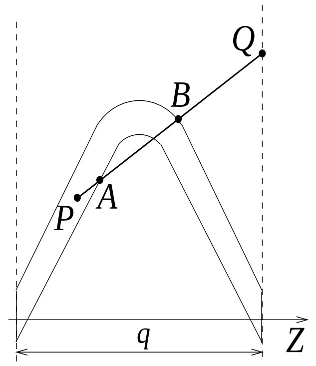

Figure 5:

The function uses an iteration technique of dividing a segment in half. At the ith step

$$\vec{P}_{i+1} = \frac{\vec{P}_i + \vec{Q}_i}{2}, \vec{Q}_{i+1} = \vec{Q}_i,$$

if the point $\frac{\vec{P}_i + \vec{Q}_i}{2}$ is not outside the fan, and

$$\vec{P}_{i+1} = \vec{P}_i, \vec{Q}_{i+1} = \frac{\vec{P}_i + \vec{Q}_i}{2},$$

otherwise. The process stops when $|\vec{P}_i - \vec{Q}_i| <$ `IterationPrecision` or i has reached the `IterationsLimit`.

When the situation is like that shown in Fig. 5, the function may either find point A or point B. It is up to the user to spot the potential ambiguity and call this function again.

The function returns the distance from the point P to one of the exiting points.

4.14 `distance_to_out`$(\vec{p}, \vec{v})$

The function calculates the distance along the vector $\vec{v}$ from the point p to the surface of the vertical fan.

First, the function determines in which `FanSection` the point p is located and finds point a, the exit point of the line $\vec{p} + \vec{v}t$ out of the `FanSection`. If a is outside of the vertical fan then the function `out_iteration_process` is called and the line's exit point q out of the fan is found. Due to the potential ambiguity of the `out_iteration_process` result (described above), it is not safe to assume that the point q is the nearest exit point. If point a is positioned inside the fan then q is set equal to a.

Next there is a search for the most remote point of segment pq from the neutral fibre. The search is performed by successively dividing the segment in half and looking for the point where the second power of the distance d from the segment pq to the neutral fibre is maximal. At the ith step

$$\vec{p}_{i+1} = \frac{\vec{p}_i + \vec{q}_i}{2}, \vec{q}_{i+1} = \vec{q}_i,$$

if the sign of the derivative $(d^2)'$ in the point $\frac{\vec{p}_i+\vec{q}_i}{2}$ is positive and

$$\vec{p}_{i+1} = \vec{p}_i, \vec{q}_{i+1} = \frac{\vec{p}_i + \vec{q}_i}{2},$$

otherwise. The sign of $(d^2)'$ is defined as the sign of the expression

$$d^2(\frac{\vec{p}_i + \vec{q}_i}{2} + \vec{\delta}) - d^2(\frac{\vec{p}_i + \vec{q}_i}{2} - \vec{\delta}),$$

where

$$\vec{\delta} = \frac{\vec{q} - \vec{p}}{|\vec{q} - \vec{p}|} \times \texttt{IterationPrecision}$$

is a small vector along the segment pq. The process is stopped when $|\vec{p}_i - \vec{q}_i| <$ `IterationPrecision` or i has reached `IterationsLimit`. The process is stopped prematurely if the point $\frac{\vec{p}_i+\vec{q}_i}{2}$ is situated outside the fan. The function `out_iteration_process`$(\vec{p}, \frac{\vec{p}_i+\vec{q}_i}{2})$ is called in this case, and its result is returned.

If the process has not been stopped prematurely, that means the whole segment pq is inside the current half-wave of the fan. If the point q has been found earlier by the function `out_iteration_process` then q is the point desired, and the function returns the length of the segment pq. Otherwise the search for the exit point is performed in the neighbor half-wave of the fan in the direction of the segment pq. The search transition is not performed if pq points beyond the beginning of the first half-wave or beyond the end of last half-wave of the fan. In that case the function returns the length of the segment pq.

Due to the properties of the geometry the search for the exiting point in the neighbor half-wave could be simplified. The point m could be determined where the line $\vec{q} + \vec{v}t$ crosses the neighbor `FanSection`, and the distance from p to the existing point could be determined from `out_iteration_process`$(\vec{p}, \vec{m})$.

4.15 `DistanceToOut`$(\vec{p}, \vec{v}, \ldots)$

The function checks if the point p is inside the `BoundingPolycone`. If yes, then it determines inside of which fan the point p is positioned and returns the distance to the fan's surface along the vector $\vec{v}$. If the point p is not inside the two nearest fans or outside of the `BoundingPolycone`, zero is returned.

The function has three more parameters, which are substituted by "…" in the header. These parameters are `G4bool calcNorm`, `G4bool *ValidNorm` and `G4ThreeVector *n`. According to Geant4 rules, if `calcNorm` is true then a function has to set parameters `*ValidNorm` and `*n`. `DistanceToOut` always sets

*ValidNorm to false. That means the solid does not lie entirely on or behind the exiting surface[1]. The value of *n has no meaning in this case.

4.16 in_iteration_process($\vec{p}, \vec{q}$)

This function searches for the entering point of the segment pq into the fan when the points p and q are positioned on different sides of the neutral fibre. (That means the distances from point p and from point q to the neutral fibre (defined as $d(\vec{p})$ and $d(\vec{q})$ respectively) are of different signs.)

The function uses the iteration technique of dividing a segment in half. At the ith step

$$\vec{p}_{i+1} = \vec{p}_i, \vec{q}_{i+1} = \frac{\vec{p}_i + \vec{q}_i}{2},$$

if the point $\frac{\vec{p}_i + \vec{q}_i}{2}$ is positioned on a different side of the neutral fibre from point p or is not located outside of the fan, and

$$\vec{p}_{i+1} = \frac{\vec{p}_i + \vec{q}_i}{2}, \vec{q}_{i+1} = \vec{q}_i,$$

otherwise. The process stops when $|\vec{p}_i - \vec{q}_i| <$ IterationPrecision or i has reached the IterationsLimit.

The function returns the distance from the point p to the entrance point of segment pq into the fan.

4.17 search_for_nearest_point($\vec{p}, \vec{q}$)

This function searches for the point on the segment pq for which the distance d to the neutral fibre of the fan is minimal. It is assumed that the value of d is unique. The points p and q are positioned on the same side of the neutral fibre, so the signs of $d(\vec{p})$ and $d(\vec{q})$ are equal. The return value is kInfinity if the point of segment pq nearest to the neutral fibre is outside the fan; if the point is inside the fan the return value is the distance from p to the fan's surface along the segment pq.

The search is conducted by dividing the segment in half. At the ith step

$$\vec{p}_{i+1} = \frac{\vec{p}_i + \vec{q}_i}{2}, \vec{q}_{i+1} = \vec{q}_i,$$

if the sign of the derivative d' in the point $\frac{\vec{p}_i + \vec{q}_i}{2}$ is negative, and

$$\vec{p}_{i+1} = \vec{p}_i, \vec{q}_{i+1} = \frac{\vec{p}_i + \vec{q}_i}{2},$$

otherwise. The sign of d' is defined as the sign of the expression

$$d(\frac{\vec{p}_i + \vec{q}_i}{2} + \vec{\delta}) - d(\frac{\vec{p}_i + \vec{q}_i}{2} - \vec{\delta}),$$

[1]This is not absolutely correct, but is not of great importance. Probably I'll correct it in future.

where

$$\vec{\delta} = \frac{\vec{q} - \vec{p}}{|\vec{q} - \vec{p}|} \times \texttt{IterationPrecision}$$

is a small vector along the segment pq. The process is stopped when $|\vec{p}_i - \vec{q}_i| <$ `IterationPrecision` or i has reached the `IterationsLimit`. The process is stopped prematurely if the point $\frac{\vec{p}_i + \vec{q}_i}{2}$ is located on the opposite side of the neutral fibre from point p. In this case the function `in_iteration_process`$(\vec{p}, \frac{\vec{p}_i + \vec{q}_i}{2})$ is called and the result returned.

If the process has not been stopped prematurely and the resulting point is outside the fan then the whole segment pq is outside of the fan; the function returns `kInfinity`. If the resulting point is inside the fan then the function `in_iteration_process` is called, and the result returned.

4.18 `distance_to_in`$(\vec{p}, \vec{v}, n_{half\_wave})$

This function determines the distance from the point p to the fan's surface along the vector $\vec{v}$. The initial search for the point of entrance into the fan is conducted in the half-wave number n_{half_wave}. The search algorithm is as follows:

The point q is found, where q is the intersection of the line $\vec{p} + \vec{v}t$ with the surface of `FanSection[`n_{half_wave}`]`. If the points p and q are positioned on the different sides of the fan's neutral fibre then the intersection of $\vec{p} + \vec{v}t$ and the fan is certainly between the boundaries of the current half-wave; the function `in_iteration_process`$(\vec{p}, \vec{q})$ is called, and its result is returned. Othewise the function `search_for_nearest_point`$(\vec{p}, \vec{q})$ is called. If its result is `kInfinity` then there is no intersection in the current half-wave; otherwise the result is the distance desired and the function returns it.

If there is no intersection with the current half-wave, then the search is moved to the neighbor half-wave along the direction of the vector $\vec{v}$. If $\vec{v}$ points beyond the beginning of the first half-wave or beyond the end of the last half-wave, the search is not moved and the function returns `kInfinity`.

Due to the properties of the geometry, the search for the entrance point could be localized to the nearest neighbor half-wave (the next half-waves are screened by neighbor fans). Let i be he point of intersection of the line $\vec{q} + \vec{v}t$ and the neighbor `FanSection`. If point i and point p are positioned on different sides of the fan, then the function `in_iteration_process` is called, and its result returned. Otherwise the function `search_for_nearest_point` is called, and its result returned.

4.19 `DistanceToIn`$(\vec{p}, \vec{v})$

The function checks if the point p is inside the `BoundingPolycone`. If yes, then it determines the distances along the vector $\vec{v}$ to the surfaces of the two fans nearest to p and returns the smaller distance. Otherwise it calls the DistanceToIn$(\vec{p}, \vec{v})$ method of `BoundingPolycone`.

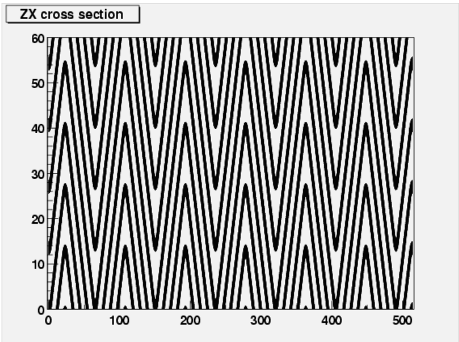

Figure 6: InnerAbsorberWheel cross-section.

5 Some test results

5.1 Tests of methods `Inside` and `SurfaceNormal`

To verify if the method `Inside` is correct the program `cross_section` has been used which finds the cross-section of a solid with the plane $y = Y_0$. Points on the plane are checked in a loop on x and z with a defined step. If the function `Inside` gives that the point is not outside the solid, then the point is marked on the 2-dim. histogram. An example histogram is shown in Fig. 6.

Looking at these cross-sections, the function `Inside` seems to be correct.

The method `SurfaceNormal` has been verified in a similar way. For this purpose the function `Inside` was temporary modified to use `nearest_point_on_neutral_fibre` instead of `distance_to_neutral_fibre`. The resulting image is similar to the one above.

5.2 Tests of methods `DistanceToIn`$(\vec{p}, \vec{v})$, `DistanceToOut`$(\vec{p}, \vec{v})$

To verify that the methods `DistanceToIn`$(\vec{p}, \vec{v})$ and `DistanceToOut`$(\vec{p}, \vec{v})$ are correct the program `distance_test` was used; this program finds the cross-section of the solid in the plane $y = Y_0$. The program scans the distance from the point p given by user to the solid along the vector $\vec{v}$. The vector $\vec{v}$ is rotated in the plane around the point p. The program draws a point in the direction $\vec{v}$ at the distance measured from p. By scanning across different points p it is possible to draw outlines of the fans.

The example of an image drawn by the program is given in Fig. 7. Looking at this image and similar ones, the functions seem to be correct.

5.3 Test using a conventional Geant4 program

The program `LArWheel` was used to verify if the new solid interacts with the internal Geant4 tools correctly. A single calorimeter end-cap was described in

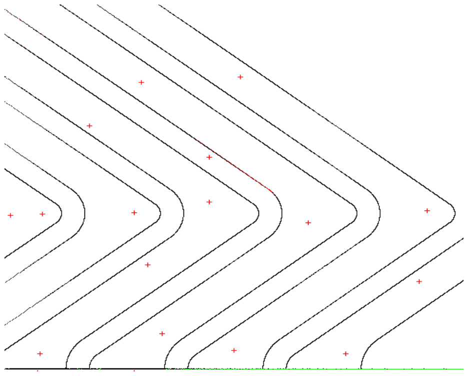

Figure 7: InnerAbsorberWheel structure, drawn using `DistanceToIn`($\vec{p}$, $\vec{v}$) and `DistanceToOut`($\vec{p}$, $\vec{v}$).

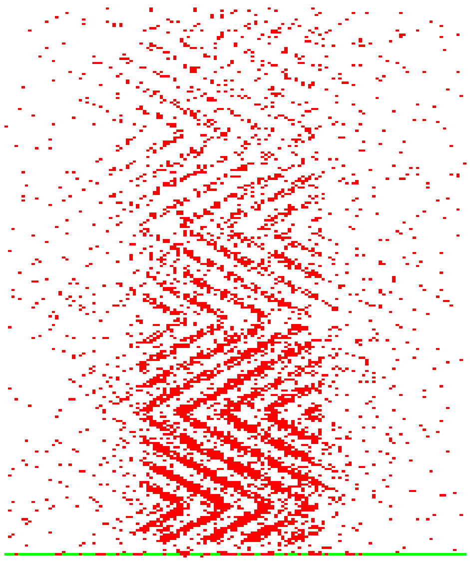

Figure 8: Low-energy photons conversion points in InnerAbsorberWheel.

the program.

The technique is: if a beam of low-enegrgy photons is directed to the solid, then the most of the photons should be converted in the absorbers. Then, by drawing the conversion points, we should see the fan-like structure of the absorbers.

The result of the test is given in Fig. 8. The waves of the absorbers are clearly seen. This demonstrates `LArWheelSolid` is able to work inside a conventional Geant4 program. A similar test with a homogenous magnetic field along the Z axis gave the same results.

An additional result of this test is a measurement of the `LArWheelSolid` routines' performance. It took about 20 minutes to process 20,000 10-MeV photons.

6 Conclusion

`LArWheelSolid` Version 0.1.0 is available on the net at:
http://cern.ch/asoukhar/LArWheelSolid-0.1.0.tar.gz.

It looks like `LArWheelSolid` is ready for serious tests in real simulation programs. But before it is released for common use, it is necessary at least to make the approximate sizes in the code exact and to check if the other sizes are really correct.

In my work on the `LArWheelSolid` the following people helped me very much: M. Leltchouk — general direction and problem-solving, S. Simion — examples of programs and useful advice, A. Chekhtman — real calorimeter geometry, V. Grichine and V. Ivanchenko — consulting on Geant4, W. Seligman — editorial assistance.

References

[1] *http://wwwinfo.cern.ch/asd/geant4/* — Geant4 Homepage.

[2] ATLAS Technical Design Report, Ch. 7 — *"The electromagnetic end-cap calorimeter and presampler"*.

[3] A. Chekhtman, D. Fouchez, E. Monnier. *"The Accordion in the end-cap: geometry and characteristics"*, ATALS-LARG-NO-4.

[4] S. Klimenko, Yu. Tikhonov, A. Chekhtman. *"The Design of Endcap EM Calorimeter with Constant Thickness of the Absorber Plates"*, ATLAS-LARG-NO-025.

[5] O. Martin, E. Monnier, S. Tisserant. *"Update of some Geometrical Parameters for the ATLAS E.M. End-Cap Calorimeter"*, ATALS-LARG-NO-047.

[6] L. Martin, J.-L. Gimenez, A. Chekhtman. *"Creating IGES files of absorbers"* (ABS.YYY.00.DRa.3).

[7] Program `white_may2000.f` written by S. Simion.

[8] LArHits Version 0.1 — *http://www.nevis.columbia.edu/~atlas/lar/*, *ftp://www.nevis.columbia.edu/pub/atlas/seligman/LArHits-0.1.tar.gz*

[9] Geant4 User's Guide — For Toolkit Developers, sec. 4 "Geometry".

Contents