Computer Physics Communications 141 (2001) 230–246

Computer Physics Communications

www.elsevier.com/locate/cpc

Mixture models of multiple scattering: computation and simulation

R. Frühwirth [a,*], M. Liendl [b]

[a] *Institute for High-Energy Physics of the Austrian Academy of Sciences, Nikolsdorfer Gasse 18, A-1050 Wien, Austria*
[b] *CERN, CH-1211 Geneva 23, Switzerland*

Received 5 June 2001

Abstract

We review the modelling of multiple Coulomb scattering by Gaussian and semi-Gaussian mixtures. We give explicit formulas for the computation of the mixture models and for the simulation from the models. We verify the convolution property of the model and compare it to GEANT4. Finally we present implementations in MATLAB and C++. © 2001 Elsevier Science B.V. All rights reserved.

Keywords: Multiple Coulomb scattering; Mixture model; Simulation

1. Introduction

Precise modelling of multiple Coulomb scattering (MS) requires an adequate representation of the tails. The tails are most prominent in thin scatterers but persist also in thicker scatterers, up to about a radiation length. In two previous publications mixture models have been described which represent the core of the projected distribution by a Gaussian and the tails either by a Gaussian [1] or by a scaled single scattering density [2]. The semi-Gaussian mixture is most suitable for the simulation of MS in thin scatterers, whereas the purely Gaussian mixture is suitable for simulation in thick scatterers and for use in track reconstruction [3]. The distinction between "thin" and "thick" scatterers is based on the approximation quality of the respective models.

We first review the two mixture models and summarize the computation of the mixture parameters. We then discuss the simulation from the models. As we focus on MS distributions far from the Gaussian limit, the projected scattering angles cannot assumed to be independent; they are, however, uncorrelated because of azimuthal symmetry if the projections are made on two orthogonal planes intersecting along the direction of the particle. In order to get the correct dependency structure of the projected angles the simulation has to be done in space. We derive the corresponding mixture models in space and give explicit formulas for generating random angles both in space and in the projections.

[*] Corresponding author.
E-mail addresses: rudolf.fruehwirth@oeaw.ac.at (R. Frühwirth), martin.liendl@cern.ch (M. Liendl).

The projected scattering angles should have the convolution property, i.e. their distribution should be independent of the number of steps taken in the scatterer. It is shown empirically that our model fulfills this requirement to very good precision. The model is also compared to the one employed by the GEANT4 simulation package. Finally, we describe the implementation of the model in MATLAB and in C++.

2. The mixture models

2.1. Physics background

In the following we always assume that the scattering angles are sufficiently small so that the projected angles can be added. Under this assumption the projected MS distributions can be obtained by repeated numerical convolution of the projected single scattering distribution. The results of this procedure have been described in detail in two previous publications [1,2]. It should be noted that the scattering angles in space cannot be added.

Using the differential cross section of elastic Coulomb scattering as given in [4], the density of the scattering angle θ in space is given by

$$f(\theta) = \frac{2a^2\theta}{(\theta^2+a^2)^2} \cdot I_{[0,b]}(\theta) \tag{1}$$

where we have assumed $\theta \ll 1$. The parameters a and b are the minimum and the maximum scattering angle, respectively [4]:

$$a = \frac{\alpha}{1.4} \cdot \frac{Z^{1/3} m_e c}{p} = \frac{2.66 \cdot 10^{-6} \cdot Z^{1/3}}{p\,[\mathrm{GeV}/c]},$$

$$b = \frac{274}{A^{1/3}} \cdot \frac{m_e c}{p} = \frac{0.14}{A^{1/3} \cdot p\,[\mathrm{GeV}/c]}. \tag{2}$$

The shape of the cutoff at $\theta = b$ is largely irrelevant; for the sake of simplicity we have used a sharp cutoff.

From the density in Eq. (1) the density of the projected single scattering angle θ_x can be derived [1]:

$$g(\theta_x) = \frac{a^2}{\pi}\left[\frac{\sqrt{b^2-\theta_x^2}}{(b^2+a^2)(\theta_x^2+a^2)} + \frac{\arccos\left(\sqrt{\theta_x^2+a^2}/\sqrt{b^2+a^2}\right)}{(\theta_x^2+a^2)^{3/2}}\right]. \tag{3}$$

Obviously both a and b depend on the material of the scatterer. If the distribution is standardized ($\sigma = 1$) the central part is nearly independent of the material, the main difference being the range of the tails (Fig. 1).

The projected MS densities can then be obtained by repeated convolutions of the projected single scattering densities. A couple of them are shown in Fig. 2.

In practice the average number $\overline{N}$ of scattering processes has to be related to the material and to the thickness of the scatterer. Assume that d is the thickness of the scatterer and X_0 is its radiation length, where d and X_0 are both given either in g/cm^2 or in cm. Let $d' = d/X_0$. Then

$$\overline{N}(d) = \frac{d\,\overline{N}_0}{X_0} = d'\overline{N}_0 \approx \frac{d' \cdot 1.587 \cdot 10^7 \cdot Z^{1/3}}{\beta^2 (Z+1)\ln(287 Z^{-1/2})} \approx \frac{d \cdot 2.215 \cdot 10^4 \cdot Z^{4/3}}{\beta^2 A}, \tag{4}$$

where $\overline{N}_0$ is the average number of scattering processes per radiation length, Z is the charge of the nucleus, A is the atomic mass number of the material and $\beta = v/c$ is related to the velocity of the scattered particle [1]. We also assume that the scattered particle has unit charge.

In what follows, all approximating mixtures are given in standard measure. The mixture has therefore to be scaled with the total standard deviation $\sigma_{\mathrm{T}}(\theta_x|d)$ of the projected MS distribution [5]:

$$\sigma_{\mathrm{T}}(\theta_x|d) = \frac{0.015}{\beta p}\sqrt{\frac{d}{X_0}}, \tag{5}$$

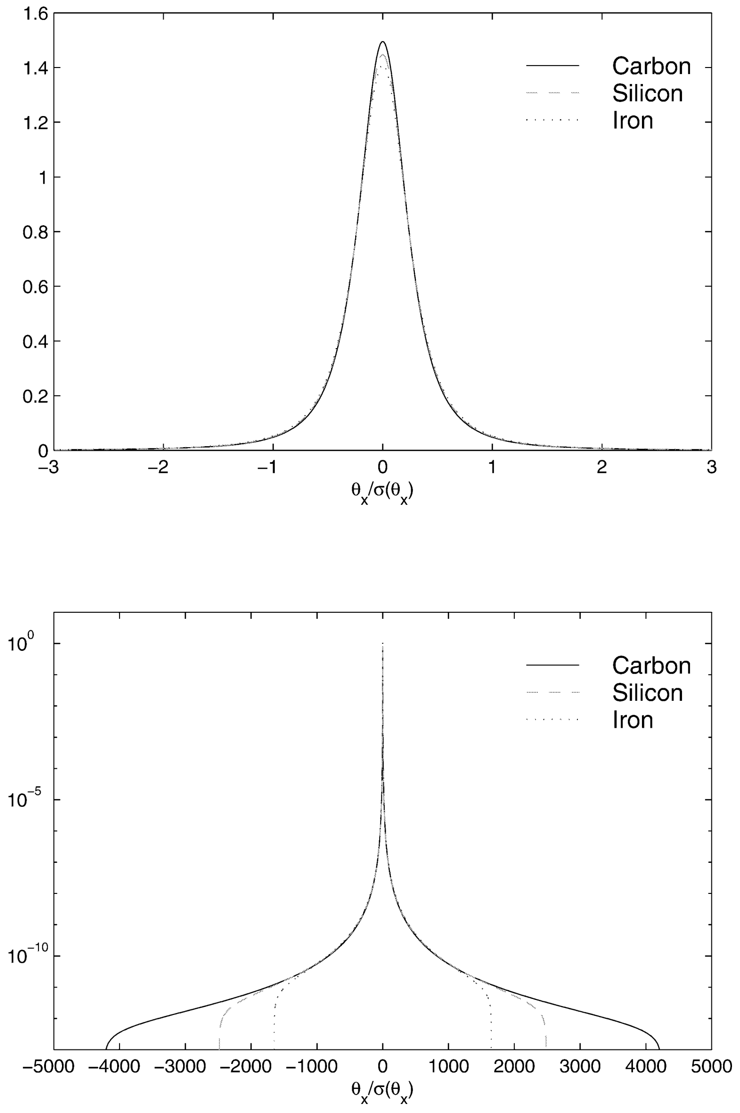

Fig. 1. The probability density function of the projected single scattering angle in standard measure. Top: central part on a linear scale, bottom: entire distribution on a logarithmic scale.

where p is the momentum of the scattered particle in GeV/c. In the implementation described below, an additional scale factor allows fine-tuning of the width by the user.

2.2. The semi-Gaussian mixture

We shall see below that in thin scatterers a Gaussian mixture is but a poor approximation to the MS distribution. For simulation purposes we propose to use a mixture with a Gaussian core and non-Gaussian tails [2]:

$$f(\theta_x) = (1-\varepsilon)\cdot\varphi\big(\theta_x;0,\sigma^2\big) + \varepsilon\cdot g(\theta_x;a,b), \tag{6}$$

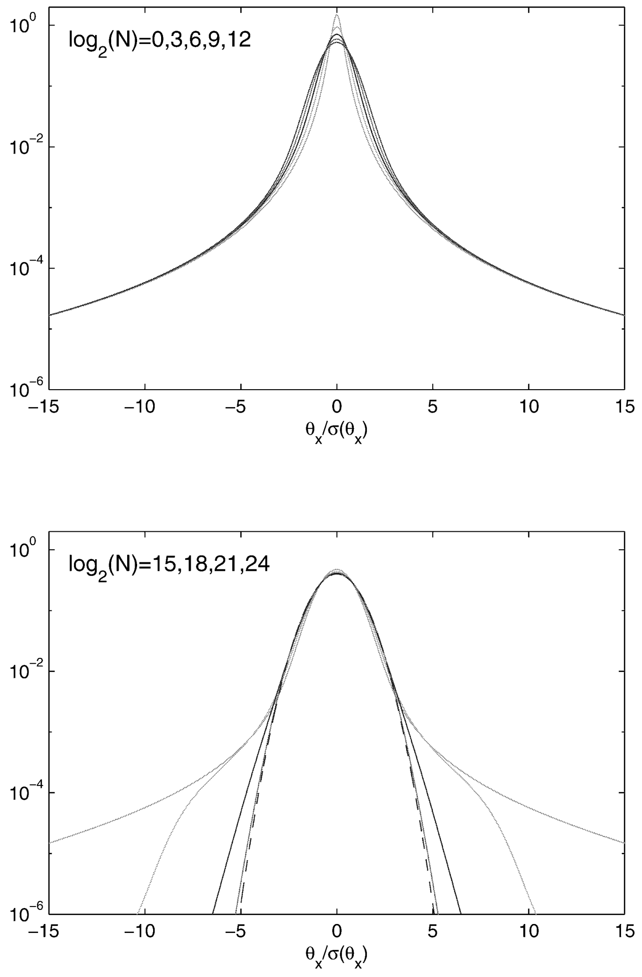

Fig. 2. Probability density functions of the projected MS angle in carbon, for N single scatters ($N = 2^n$, $n = 0, 3, 6, \ldots, 21, 24$). The dashed line in the bottom figure is the density of a standard Gaussian. All densities are in standard measure.

where $\varphi(\theta_x; 0, \sigma^2)$ is the Gaussian density with mean 0 and variance σ^2, and $g(\theta_x; a, b)$ is the density of the projected single scattering angle (see Eq. (3)).

This mixture model has four free parameters $(a, b, \varepsilon, \sigma^2)$. In standard measure the tail weight ε can be expressed in terms of a, b, and σ^2:

$$\varepsilon = \max\left(0, \frac{1-\sigma^2}{a^2[\ln(b/a) - 0.5] - \sigma^2}\right). \tag{7}$$

Above a certain thickness of the scatterer this mixture model is no longer appropriate, and ε would become negative. In this case it is set to 0. If this happens, the Gaussian mixture model should be used instead.

The parameter b can be approximated by

$$b = \frac{\varrho}{\sqrt{N}(\ln\varrho - 0.5)}, \quad \text{with } \varrho = \frac{b}{a} \approx \frac{41000}{Z^{2/3}}. \tag{8}$$

A nearly material-independent representation in the required range can be achieved if the core variance σ^2 and the tail parameter a are considered as functions of $n = Z^{0.1}\ln\overline{N}$, where $\overline{N}$ is the average number of scattering processes (see Eq. (4)). In [2] a polynomial approximation of degree 2 and 4, respectively, has been obtained:

$$\sigma^2 = 1.827\cdot 10^{-1} + 3.803\cdot 10^{-2}\cdot n + 5.783\cdot 10^{-4}\cdot n^2, \tag{9}$$

$$a = 2.822\cdot 10^{-1} + 9.828\cdot 10^{-2}\cdot n - 1.355\cdot 10^{-2}\cdot n^2 + 1.330\cdot 10^{-3}\cdot n^3 - 4.590\cdot 10^{-5}\cdot n^4. \tag{10}$$

2.3. The Gaussian mixture

The purely Gaussian mixture used in [1] has the form

$$f(\theta_x) = (1-\varepsilon)\cdot\varphi\big(\theta_x; 0, {\sigma_1}^2\big) + \varepsilon\cdot\varphi\big(\theta_x; 0, {\sigma_2}^2\big), \tag{11}$$

where ${\sigma_1}^2 < {\sigma_2}^2$ and $\varepsilon < 0.5$. The core variance ${\sigma_1}^2$ is parameterized in terms of the reduced thickness $d_0' = d/\beta^2 X_0$, where X_0 is the radiation length of the scatterer:

$${\sigma_1}^2 = 8.471\cdot 10^{-1} + 3.347\cdot 10^{-2}\cdot\ln d_0' - 1.843\cdot 10^{-3}\cdot\big(\ln d_0'\big)^2. \tag{12}$$

The tail weight ε is parameterized in terms of a modified reduced thickness $d_0'' = Z^{2/3}\,d/\beta^2 X_0$:

$$\varepsilon = \begin{cases} 4.841\cdot 10^{-2} + 6.348\cdot 10^{-3}\cdot\ln d_0'' + 6.096\cdot 10^{-4}\cdot\big(\ln d_0''\big)^2, & \text{if } \ln d_0'' < 0.5,\\ -1.908\cdot 10^{-2} + 1.106\cdot 10^{-1}\cdot\ln d_0'' - 5.729\cdot 10^{-3}\cdot\big(\ln d_0''\big)^2, & \text{if } \ln d_0'' \geqslant 0.5. \end{cases} \tag{13}$$

As the mixture is in standard measure the tail variance can be computed by

$${\sigma_2}^2 = \frac{1-(1-\varepsilon){\sigma_1}^2}{\varepsilon}. \tag{14}$$

If $d/\beta^2 X_0$ is larger than 10, the tail component of the Gaussian mixture is negligible, and a single Gaussian is sufficient.

2.4. Comparison with the actual distributions

The quality of the approximation can be assessed by the Kullback–Leibler divergence $D(f\,||h)$ of the mixture $f(\theta_x)$ from the actual MS density $h(\theta_x)$ as obtained by convolution:

$$D(f\,||h) = \int f(\theta_x)\ln\big(f(\theta_x)/h(\theta_x)\big)\,\mathrm{d}\theta_x. \tag{15}$$

Fig. 3 shows the Kullback–Leibler divergence of the semi-Gaussian mixture (6), the Gaussian mixture (11), and the Gaussian approximation with the Highland formula [6,7]. It can be seen that the Gaussian mixture is always better than the purely Gaussian model, and that below a certain thickness the semi-Gaussian mixture is better than the Gaussian mixture. The point of crossover can be parameterized by $d/\beta^2 X_0 = 0.6/Z^{0.6}$.

Figs. 4 and 5 show the actual and the model densities for some values of d/X_0. We assume here and in what follows that $\beta = 1$. It is obvious that for thin scatterers the purely Gaussian mixture is a poor approximation to the actual distribution, whereas the semi-Gaussian mixture fits almost perfectly.

3. Simulation

If θ is the MS angle in space, the projected MS angles $\theta_x = \theta\cos\varphi$ and $\theta_y = \theta\sin\varphi$ are always uncorrelated, because of the azimuthal symmetry. They are, however, not independent, except in the Gaussian limit where their

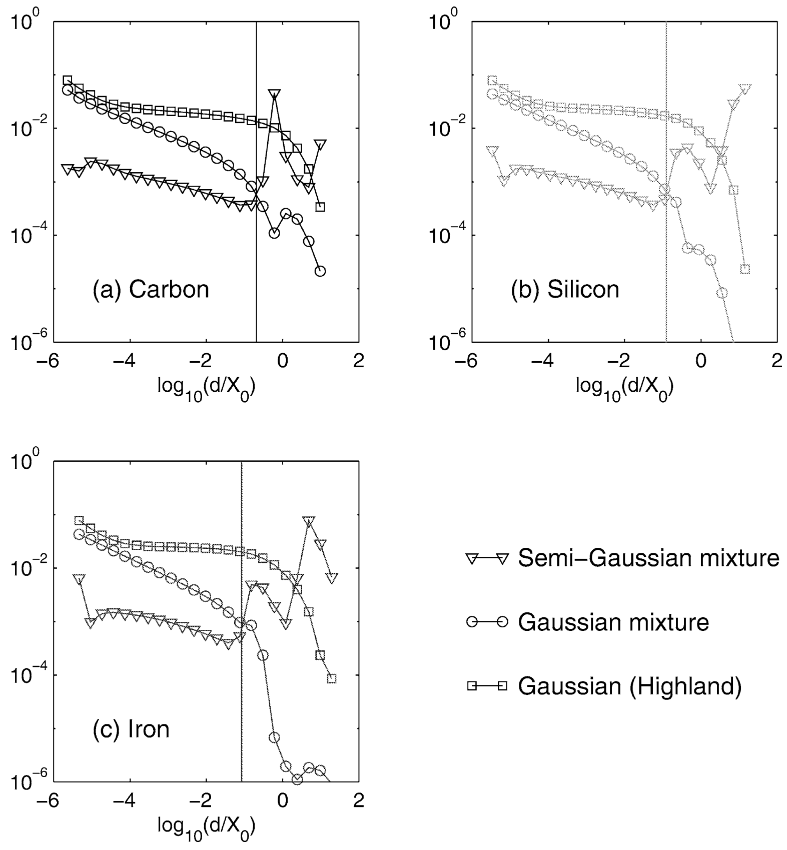

Fig. 3. The Kullback–Leibler divergence of the two mixture models and the Gaussian model due to Highland, as a function of the number of radiation lengths traversed. The vertical line shows the point of crossover between the mixture models.

joint distribution is a two-dimensional Gaussian. In order to get the correct dependency structure when simulating pairs of projected scattering angles, one has to go back to the spatial scattering angle θ. Although the distribution of $\theta = (\theta_x^2 + \theta_y^2)^{1/2}$ is in general not uniquely determined by the projected distributions of θ_x and θ_y, it is not difficult to find a plausible mixture density of θ which has the desired projected mixture densities of θ_x and θ_y.

3.1. *Simulation from the single scattering distribution*

In very thin layers the best precision is obtained when multiple scattering is simulated by summing over a random number of single scatters, the number being distributed according to a Poisson distribution. We propose to apply this procedure whenever the average number of scatters is below 10. The single scattering angle in space θ has the density given in Eq. (1) and is generated according to the formula

$$\theta = ab\sqrt{\frac{1-u}{ub^2+a^2}}, \tag{16}$$

where u is uniformly distributed in $[0, 1]$ and a, b are determined according to Eq. (2). The azimuth φ is drawn from a uniform distribution in $[0, 2\pi]$, independently of u.

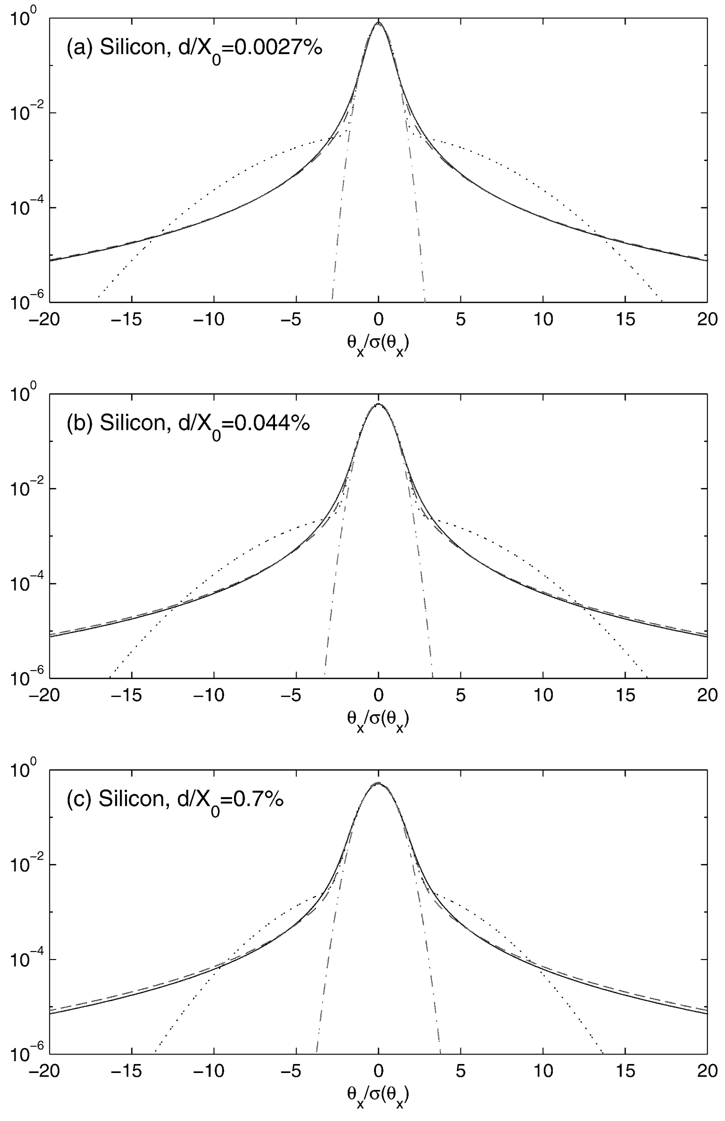

Fig. 4. The probability density function of the projected MS angle in silicon obtained by convolution (solid line), the semi-Gaussian mixture model (dashed line), the Gaussian mixture model (dotted line), and the Gaussian model due to Highland (dashed-dotted line). All densities are standardized.

3.2. Simulation from the semi-Gaussian mixture

It can easily be shown that the following mixture model of the distribution of θ yields projected densities of the desired form (6):

$$f(\theta) = (1-\varepsilon)\cdot\frac{\theta\exp(-\theta^2/2\sigma^2)}{\sigma^2} + \varepsilon\cdot\frac{2a^2\theta}{(\theta^2+a^2)^2}\cdot I_{[0,b]}(\theta), \tag{17}$$

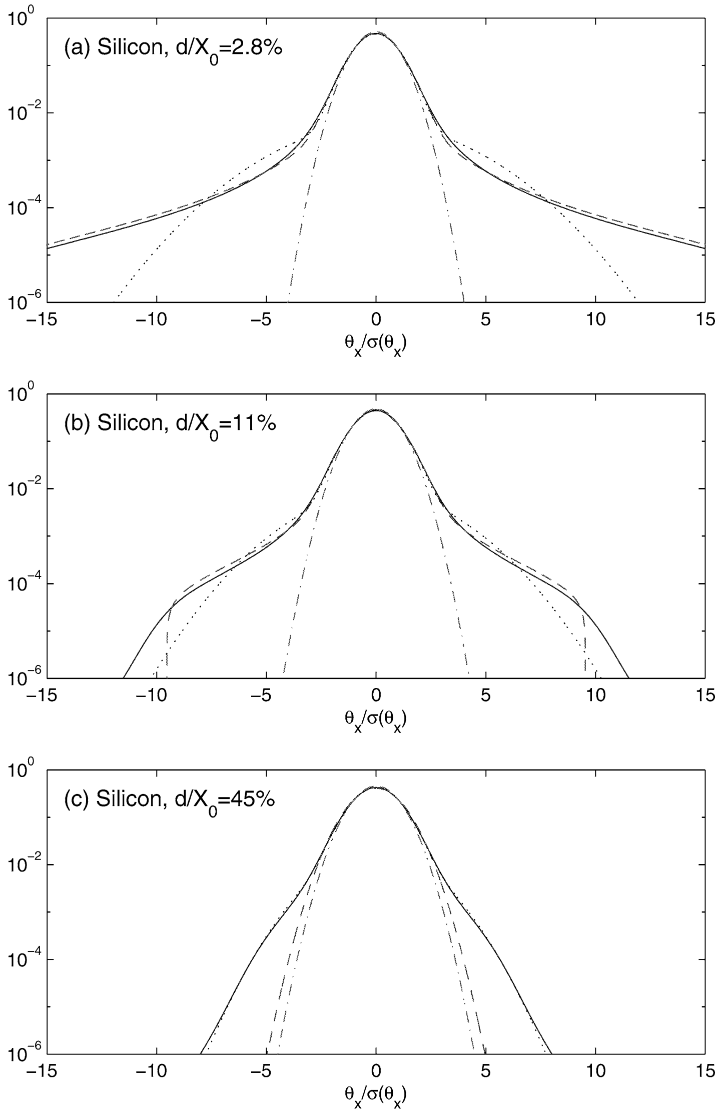

Fig. 5. The probability density function of the projected MS angle in silicon obtained by convolution (solid line), the semi-Gaussian mixture model (dashed line), the Gaussian mixture model (dotted line), and the Gaussian model due to Highland (dashed-dotted line). All densities are standardized.

where the second component is the density of the single scattering angle in space (Eq. (1)). Drawing a random number from the distribution with density (17) is accomplished in the following way:

$$\theta = \begin{cases} \sigma\sqrt{-2\ln u} & \text{if } \ v > \varepsilon, \\ ab\sqrt{\dfrac{1-u}{ub^2+a^2}} & \text{if } \ v < \varepsilon, \end{cases} \tag{18}$$

where u and v are independent and uniformly distributed in $[0, 1]$. The azimuth φ is drawn from a uniform distribution in $[0, 2\pi]$, independently of u and v.

3.3. Simulation from the Gaussian mixture

The following mixture model of the distribution of θ yields projected mixture densities of the desired form (11):

$$f(\theta) = (1-\varepsilon)\cdot\frac{\theta\exp(-\theta^2/2{\sigma_1}^2)}{{\sigma_1}^2} + \varepsilon\cdot\frac{\theta\exp(-\theta^2/2{\sigma_2}^2)}{{\sigma_2}^2}. \tag{19}$$

Drawing a random number from the distribution with density (19) is accomplished in the following way:

$$\theta = \begin{cases}\sigma_1\sqrt{-2\ln u} & \text{if } v > \varepsilon,\\ \sigma_2\sqrt{-2\ln u} & \text{if } v < \varepsilon,\end{cases} \tag{20}$$

where u and v are independent and uniformly distributed in $[0, 1]$. The azimuth φ is drawn from a uniform distribution in $[0, 2\pi]$, independently of u and v.

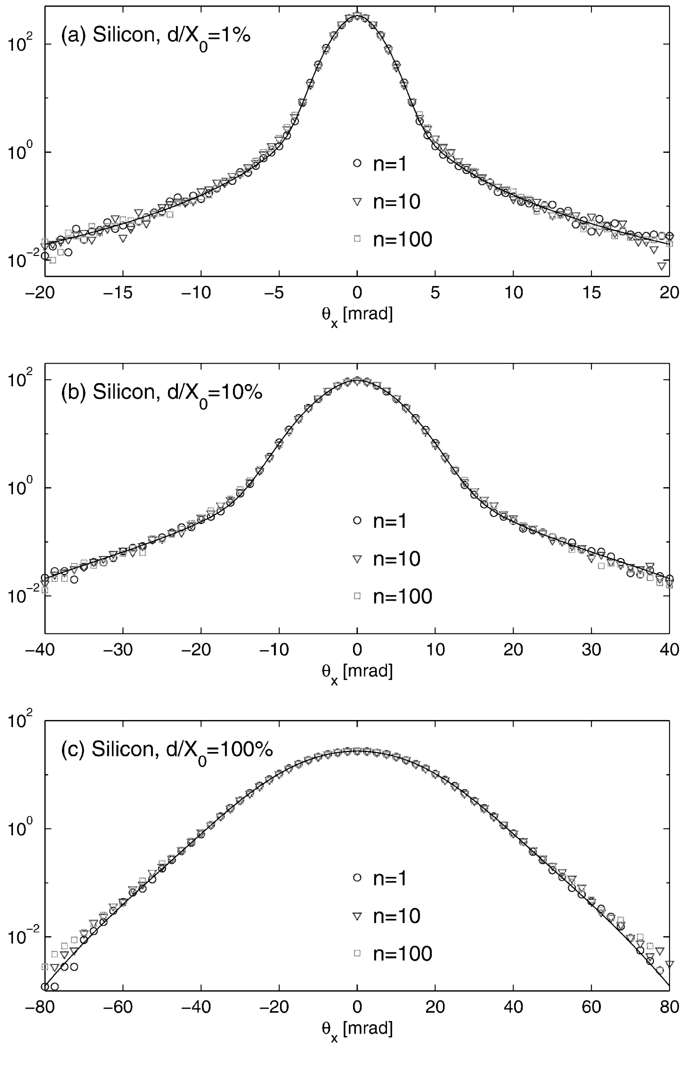

Fig. 6. The probability density function of the projected MS angle in a silicon scatterer of thickness d (solid line) and the frequency distributions of simulated samples obtained by dividing the scatterer into n slices ($n = 1$: circles, $n = 10$: triangles, $n = 100$: squares).

4. The convolution property

A statistical model of multiple scattering in a projection should have the convolution property: The scattering angle induced by a scatterer of thickness d should have the same distribution as the convolution of n scattering angles induced by n contiguous scatterers of thickness d/n each. In a simulation program the lack of the convolution property could lead to some unpleasant surprises. The convolution property is trivially fulfilled by the Gaussian model if the variance is strictly proportional to the thickness d. In the Highland model [6,7] the convolution property is violated, albeit to a fairly small degree. If we consider, for instance, a scatterer with a thickness of 0.1 radiation lengths, the variance of the projected scattering angle according to Highland is equal to $\sigma^2 = 15.4/(p\beta)^2$, p being given in MeV$/c$. If the same scatterer is considered as an assembly of ten scatterers of

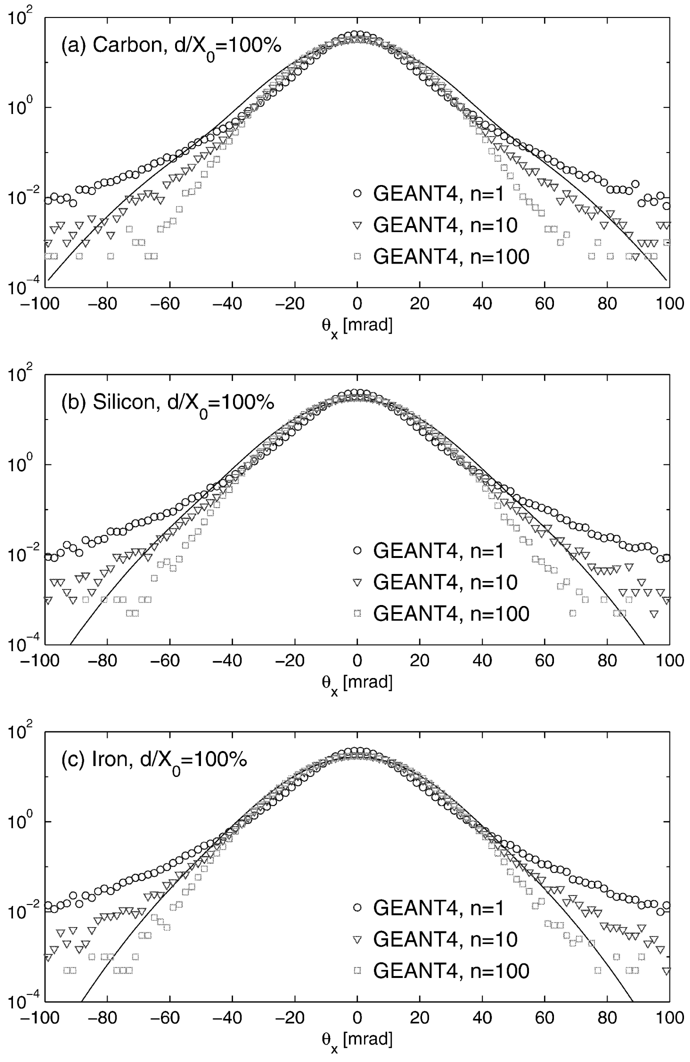

Fig. 7. The probability density function of the projected MS angle in three scatterers of thickness $d = X_0$ (our model, solid line), and the frequency distributions of simulated GEANT4 samples obtained by dividing the scatterer into n slices ($n = 1$: circles, $n = 10$: triangles, $n = 100$: squares).

0.01 radiation lengths each, the variances sum up to $\sigma^2 = 12.6/(p\beta)^2$ or about 82 percent of the value above. This reflects the fact that the Highland model deliberately neglects the tails and models only the width of the Gaussian core. This feature of the Highland model has been our initial motivation to look for models with strictly additive variances which nevertheless reproduce the width of the core.

As our model is a rather simple approximation to the actual distributions we cannot expect that the convolution property holds precisely as far as the shape of the distributions is concerned, although the model is constructed such that the variances are always additive. We can, however, check empirically to which degree the shape of the distribution obeys the convolution property. We give a few examples with silicon scatterers. Fig. 6 shows the mixture density of the projected scattering angle after 1 percent, 10 percent and 100 percent of a radiation length in silicon, the distribution of the corresponding simulated sample, and the distributions of simulated samples obtained by dividing the scatterer into 10 and 100 slices, respectively. The agreement is excellent, both in the Gaussian core and in the tails.

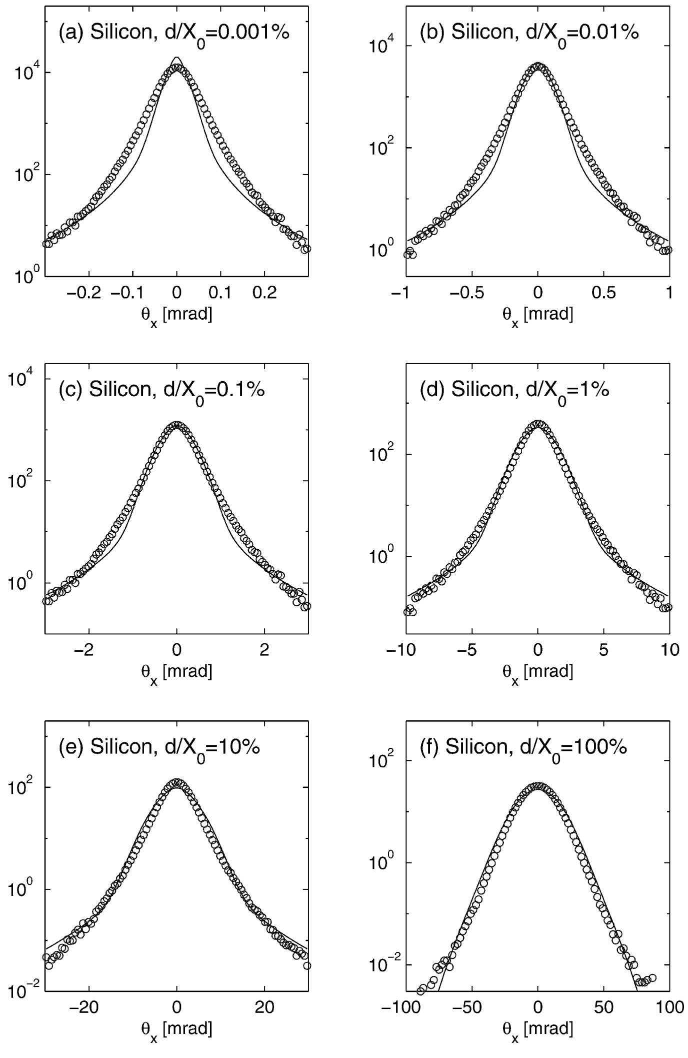

Fig. 8. The probability density function of the projected MS angle in a silicon scatterer of thickness d (our model, solid line) and the frequency distributions of simulated GEANT4 samples (circles). For $d = X_0$, the scatterer has been divided into 10 slices.

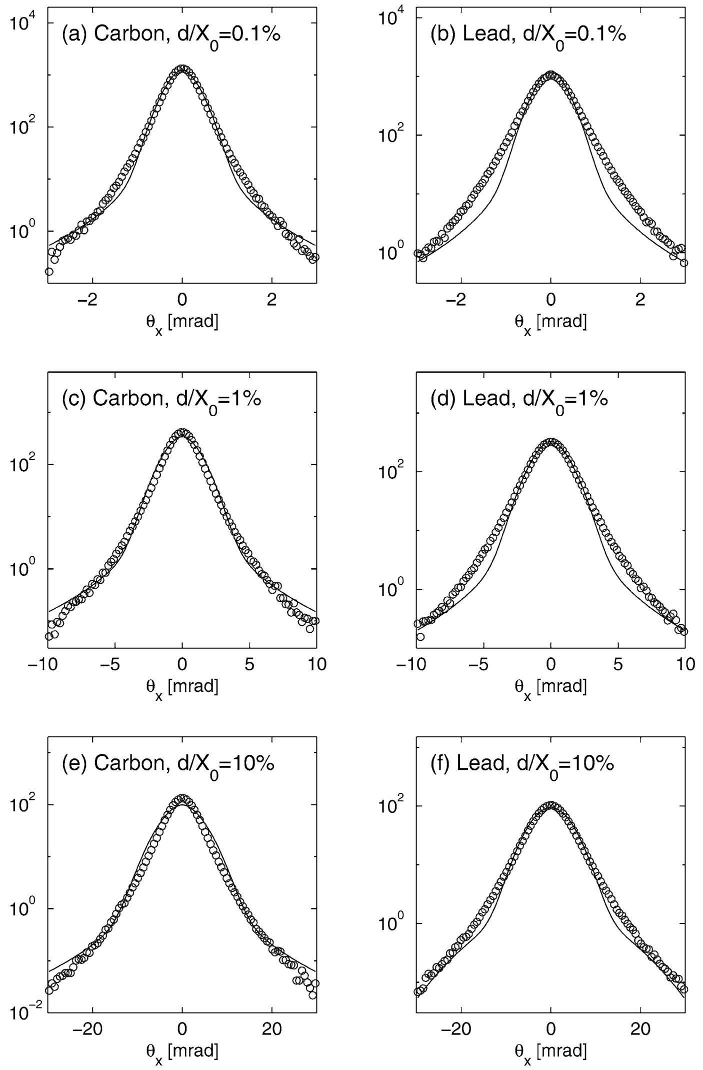

Fig. 9. The probability density function of the projected MS angle in a scatterer of thickness d (our model, solid line) and the frequency distributions of simulated GEANT4 samples (circles). The material is carbon (left hand side) and lead (right hand side).

Table 1
Comparison of computation time between MATLAB and C++

Model	d/X_0	C++ [sec]	MATLAB [sec]	MATLAB/C++
GaussMix	2	16.4	44.2	2.7
SemiGauss	$1 \cdot 10^{-4}$	15.7	47.9	3.1
FewScatt[a] ($\overline{N} = 8.6$)	$1 \cdot 10^{-5}$	105.4	535.3	5.1
FewScatt ($\overline{N} = 4.3$)	$5 \cdot 10^{-6}$	56.7	279.3	4.9
FewScatt ($\overline{N} = 0.86$)	$1 \cdot 10^{-6}$	17.2	92.0	5.3

[a] Simulation by summing over a Poisson distributed number of single scatters.

5. Comparison with GEANT4

In this section we compare our model with the distributions produced by GEANT4 [8], which is a very general simulation program widely used in high-energy and nuclear physics. Version GEANT4.2.0.ref03 was used to produce the results presented here. No significant differences in the general behaviour of the GEANT4 multiple scattering model were found when comparing selected results with the latest available version GEANT4.3.1.

The comparison is complicated by the fact that the model currently employed by GEANT4 lacks the convolution property. As an example, Fig. 7 shows the frequency distributions of the simulated projected scattering angle after a radiation length of material, the material being divided into 1, 10, and 100 slices. They exhibit discrepancies both in the core and in the tails of the distribution. Neither of them agrees extremely well with our model. For carbon and silicon, the best agreement is for $n = 10$, while for iron it is best for $n = 100$.

Fig. 8 shows a comparison of our model with the GEANT4 simulations, over a wide range of the thickness of the scatterer. The material is silicon. The agreement is good for scatterers thicker than about one percent of a radiation length, and is somewhat worse for thinner scatterers. For $d = X_0$, we have chosen the sample with $n = 10$ slices in order to get better agreement. Fig. 9 shows more comparisons, the material being carbon and lead. The agreement is rather good for carbon, and clearly worse for lead.

Figs. 8 and 9 also show that in GEANT4 the shape of the distribution changes very little with the thickness of the scatterer, whereas in our model the change is much more pronounced. This, in conjunction with the fact that the GEANT4 model lacks the convolution property, leads us to suspect that the current GEANT4 model is still capable of improvement.

6. Implementation

We have implemented the model in MATLAB [9] and C++. Although the C++ version was not designed for numerically optimal performance, it is still faster than the vectorized MATLAB version by a factor between 3 and 5. Table 1 shows a speed comparison between the two implementations.

The simulations were carried out on a Pentium III 600 MHz PC, with MATLAB 5.3 running under Windows NT 4.0, and the C++ implementation compiled with GNU egcs-2.91.66 and running under Linux Redhat 6.1. We have simulated 10^7 events for 1 GeV$/c$ muons in carbon varying the thickness in order to trigger different models.

We now give a brief description of both versions. The source code is available from the authors on request.

6.1. MATLAB version

There are two main functions which compute the mixture density and generate scattering angles, respectively. In both functions the scale factor `scale` can be used to adjust the standard deviation of the distribution. It should be set to 1 if no scaling is required.

The mixture density can be computed by calling the function `get_density`. The calling sequence and the explanation of the input and output parameters is shown in Table 2.

Two vectors of projected scattering angles can be generated by calling the function `sim_mixture`, described in Table 3.

In addition to the two main functions, there is a set of internal functions which are called by the two main functions, but can also be called by the user (Table 4).

Note that the functions `gaussmix_pdf` and `semigauss_pdf` return the standardized probability density function which has to be scaled by `sigma_tot`. All functions `sim_*` internally call the MATLAB function `rand` which produces uniform random number in the interval $[0, 1]$.

Table 2
The MATLAB function `get_density`

```
%-------------------------------------------------------------%
function [f,sigma_tot]=get_density(x,beta,p,d,Z,A,scale)      %
% Compute the probability density function of the mixture     %
%-------------------------------------------------------------%
% Input: x ...... vector of x values                          %
%        beta ... v/c (0<beta<=1)                             %
%        p ...... momentum in GeV/c                           %
%        d ...... thickness of scatterer over radiation length %
%        Z ...... average nuclear charge                      %
%        A ...... average atomic mass                         %
%        scale .. scale factor on total standard deviation    %
%-------------------------------------------------------------%
% Output: f ......... probability density function f(x)       %
%         sigma_tot.. (scaled) standard deviation             %
%-------------------------------------------------------------%
```

Table 3
The MATLAB function `sim_mixture`

```
%-------------------------------------------------------------%
function [thx,thy]=sim_mixture(beta,p,d,Z,A,scale,n)          %
% Simulate two arrays of uncorrelated scattering angles       %
%-------------------------------------------------------------%
% Input: beta ... v/c (0<beta<=1)                             %
%        p ...... Momentum in GeV/c                           %
%        d ...... Thickness of scatterer over radiation length %
%        Z ...... average nuclear charge                      %
%        A ...... average atomic mass                         %
%        scale .. scale factor on total standard deviation    %
%        n ...... length of thx and thy                       %
%-------------------------------------------------------------%
% Output: thx,thy ... two scattering angles                   %
%                     in orthogonal projections               %
%                     thx and thy are of size (n,1)           %
%-------------------------------------------------------------%
```

6.2. C++ version

An UML diagram of the class design is shown in Fig. 10. The class `Msc` serves as an interface to the user. The required simulation parameters have to be filled into `MscParams`. According to the given parameters the method `Msc::configure(..)` selects the appropriate multiple-scattering model automatically. A manual selection is, however, always possible through a call to `Msc::chooseUnit(..)`. The method `Msc::simulate(thx,thy)` triggers the actual simulation of n pairs of projected scattering angles, where n is the size of `thx`, and stores them in the vectors `thx` and `thy`, respectively. The code snippet in Table 5 illustrates the usage of the C++ implementation.

Each multiple scattering model is implemented as a subclass of `MscUnit`. The subclasses `MscSingle, MscFew, MscSemiGauss, MscGaussMix` correspond to their MATLAB counterparts `sim_single-scatt, sim_fewscatt` and so on, and can be used directly without the `Msc` interface class. In order to set the

Table 4
Other MATLAB functions

```
%-----------------------------------------------------------------%
function [var1,var2,epsi,sigma_tot]=get_gaussian(beta,p,d,Z)      %
% Get the parameters of the pure Gaussian                         %
%-----------------------------------------------------------------%
function [var1,var2,epsi,sigma_tot]=get_gaussmix(beta,p,d,Z)      %
% Get the parameters of the Gaussian mixture                      %
%-----------------------------------------------------------------%
function [a,b,var1,epsi,sigma_tot,N]=get_semigauss(beta,p,d,Z)    %
% Get the parameters of the semi-Gaussian mixture                 %
% N is the average number of scattering processes                 %
%-----------------------------------------------------------------%
function [a,b,sigma]=get_single(p,Z,A)                            %
% Get the parameters of the single scattering distribution        %
%-----------------------------------------------------------------%
function y=gaussmix_pdf(x,var1,var2,epsi)                         %
% Compute the standardized density of the Gaussian mixture        %
%-----------------------------------------------------------------%
function y=semigauss_pdf(x,a,b,var1,epsi)                         %
% Compute the standardized density of the semi-Gaussian mixture   %
%-----------------------------------------------------------------%
function [thx,thy]=sim_gaussmix(var1,var2,epsi,sigma_tot,n)       %
% Simulate from the Gaussian mixture                              %
%-----------------------------------------------------------------%
function [thx,thy]=sim_semigauss(a,b,var1,epsi,sigma_tot,n)       %
% Simulate from the semi-Gaussian mixture                         %
%-----------------------------------------------------------------%
function [thx,thy]=sim_fewscatt(a,b,sigma,lambda,n)               %
% Simulate multiple scattering by summing over single scatters    %
% The multiplicity is Poisson distributed with mean lambda        %
% sigma is the standard deviation of the single scatter           %
%-----------------------------------------------------------------%
function [thx,thy]=sim_singlescatt(a,b,n)                         %
% Simulate from the standardized single scattering density        %
%-----------------------------------------------------------------%
```

required parameters the method `MscX::initMe(..)` has to be called (`X = Single`, `Few`, `SemiGauss`, `GaussMix`). `MscX::simulate(..)` triggers the simulation of the specific subclass in the same way as `Msc::simulate(..)` does for the automatically selected simulation model.

Uniformly distributed random numbers are drawn inside the subclasses of `MscUnit` with the `RandFlat` generator provided by the CLHEP [10] library.

7. Discussion

We have presented a complete mixture model of multiple scattering. The distribution of the projected scattering angle is approximated by a Gaussian mixture for thick scatterers, and by a mixture of a Gaussian core and single scattering tails for thin scatterers. In very thin scatterers, simulation is performed by adding single scatters, the multiplicity being Poisson distributed. Implementations in MATLAB and C++ are available. By default, the model is chosen automatically, based on the thickness of the scatterer; the user is free, however, to use the model of his

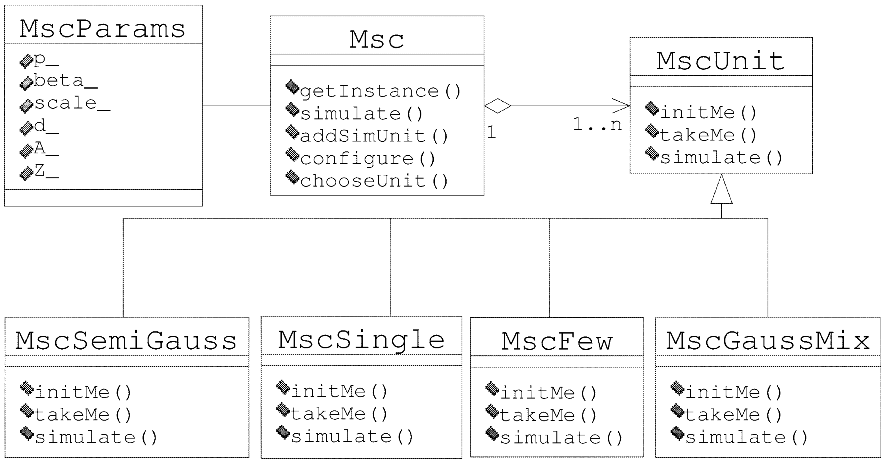

Fig. 10. UML class diagram of the C++ implementation. Class `Msc` acts as a singleton interface for the user and implements a bridge to the simulation unit `MscUnit`. A list of instances of different simulation models subclassed from `MscUnit` (`MscSingle, -Few, -SemiGauss, -GaussMix`) is managed by `Msc`. Depending on the parameters provided in `MscParams` the polymorphic method `MscUnit::takeMe(..)` decides in each subclass if the specific model should be chosen. `Msc::configure(..)` takes an instance of `MscParam` and calls each `::takeMe(..)` to choose the appropriate model. `Msc::simulate` delegates the simulation task to the chosen model.

Table 5
Usage of the C++ implementation

```
/*************************************************************/
// beta=1.v/c, p = 2.GeV/c , d=0.1 d/Xo , Z=6, A=12.01       //
// optional scale factor can be given as last argument       //
// to the constructor to adjust the total standard           //
// deviation of the coosen distribution. By default          //
// the scale factor is 1.                                    //
MscParams thePara(1., 2., 0.1, 6, 12.01);                    //
// instantiate the Msc user-interface                        //
Msc * simInterface = Msc::getInstance();                     //
// automatically select simulation model                     //
simInterface->configure(thePara);                            //
// simulate two scattering angles in orthogonal projections  //
// result saved in vector thx and thy of specified size      //
int n(10000)               // size of vectors                //
vector<double> thx(n);                                       //
vector<double> thy(n);                                       //
simInterface->simulate(thx,thy);                             //
/*************************************************************/
```

choice. Using only the Gaussian core of the mixture gives results which are very similar to the ones by using the Highland model, as shown in [1].

The model is restricted to the simulation of projected scattering angles and does not deal with the lateral displacements of the scattered particles. In any case, the lateral displacements are entirely negligible for thin scatterers. If lateral displacements are significant, we recommend to divide the scatterer into several thinner slices and to integrate the lateral displacements induced by the angular deflections. This produces the correct correlations with the angles, and has the additional advantage that other processes like energy loss can be taken into account more frequently and thus with higher accuracy.

Acknowledgement

We thank W. Mitaroff (HEPHY, Vienna) for numerous useful comments on the manuscript.

References

[1] R. Frühwirth, M. Regler, Nucl. Instrum. Methods A 456 (2001) 369.
[2] R. Frühwirth, M. Regler, Modelling of multiple scattering distributions by mixture models, Proceedings of the International Conference on Advanced Monte Carlo for Radiation Physics, Particle Transport Simulations and Applications, Lisbon, October 2000.
[3] R. Frühwirth, Comput. Phys. Commun. 100 (1997) 1.
[4] J.D. Jackson, Classical Electrodynamics, John Wiley & Sons, New York, 1999.
[5] R. Frühwirth et al., Data Analysis Techniques for High-Energy Physics, Cambridge University Press, Cambridge, 2000.
[6] V. Highland, Some practical remarks on multiple scattering, Nucl. Instrum. Methods 129 (1975) 497.
[7] Particle Data Group, C. Caso et al., Passage of Particles through Matter, in: Review of Particle Physics. European Phys. J. C 3 (1998) 144.
[8] Documents can be obtained by following links from the GEANT4 homepage at CERN (http://wwwinfo.cern.ch/asd/geant4/geant4.html).
[9] The MathWorks, Inc., MATLAB Language Reference Manual, Version 5.1, 1997.
[10] Documents can be obtained by following links from the CLHEP homepage at CERN (http://wwwinfo.cern.ch/asd/lhc++/clhep).