Revisiting the SPLITS Model: Towards an Enhanced Implementation

Mark Y. Iwanchyshyn, Bradley W. Kimmel and Gladimir V. G. Baranoski

Natural Phenomena Simulation Group, David R. Cheriton School of Computer Science, University of Waterloo, Canada

Technical Report: CS-2020-01

February 23, 2020

Abstract

The first-principles model known as SPLITS (SPectral LIght Transport model for Sand) was developed by the Natural Phenomenon Simulation Group (NPSG) at the University of Waterloo. This model has been employed in a wide range of investigations involving particulate materials. In this report, we present the refinements made to the model's implementation and introduce its enhanced version.

1 Introduction

Since the development of the SPLITS model [13, 14], it has been used in a number of investigations in computer graphics and remote sensing (*e.g.*, [1, 3, 4, 5, 6, 7, 8, 9, 13, 15, 16]). This model employs an innovative approach for the stochastic simulation of light transport in particulate materials. This approach allows the direct simulation of light (ray) interactions with specific material constituents (*e.g.*, sand grain, cells and organelles) without having to explicitly store them. Accordingly, it has been incorporated into other first-principles models, like CLBlood (Cell-based model of Light interaction with whole Blood) [17, 28] and HyLIoS (Hyperspectral Light Impingement on Skin) [10], developed by our group.

It is a well-known fact that any computer simulation code, regardless of its complexity and the programming skills of the people responsible for its implementation, is subject to the occurrence of "bugs". These apparently tiny errors (*e.g.*, a flipped minus sign) may not necessarily cause a simulation to break down, but they can have a significant impact on its results [20].

Within this context, our group has found out that it pays off to obtain a "fresh" version of a model's code. That is, have it rewritten by researchers not involved in its original formulation and implementation. However, we also noted that, if such an undertaking can be carried out with the support of those that have been involved in the development of the model, the chances of amplifying problems instead of fixing them are mitigated.

This code rewriting strategy has been systematically employed by our group [2]. It has enabled us not only to filter out possible bugs in the implementations of our models, but also to increase their fidelity to cost ratio through the use of more efficient software resources. Moreover, it has also facilitated the maintenance and the incorporation of new features to our models since the revised code tends to be structured in a more straightforward manner.

These aspects have also motivated this research project which had two main goals. The first was to revisit the model's formulation and rewrite its code from a "fresh" standpoint. The second was to release an enhanced version of its implementation, henceforth referred to as SPLITS-2, for online deployment through our model distribution framework NPSGD (Natural Phenomena Simulation Group Distributed) [2].

We began this project as a reconstruction of the model from its fundamental components using its original descriptions [13, 14] and input data (*e.g.*, spectral refractive indices and extinction coefficients [22]). This black-box approach proved to be less cost-effective than we originally expected. Accordingly, we switch to a white-box approach, which we describe in this document. This approach allowed us to identify mistakes in the original implementation when there were irreconcilable differences in the output of the testing simulations. The corrections to these mistakes were then incorporated into SPLITS-2.

The remainder of this document is organized as follows. In Section 2, we describe the enhancements to the model's implementation. In Section 3, we compare modeled results with measured data to demonstrate that the model's predictive capabilities have been preserved. In Section 4, we briefly address the online deployment of SPLITS-2. Finally, in Section 5, we summarize the main outcomes of this project.

2 Implementation Enhancements

In this section, we described specific modifications performed in the model's implementation in order to completely align it with the model's original formulation while preserving its predictive capabilities. More precisely, we identify the implementation problems, briefly explain their underlying causes and state how we fixed them. Low level code details about these problems are provided in the Appendices A to C.

2.1 Non-Fixed Soil Texture Issue

Soil samples are normally composed of particles (grains) of weathered rock immersed in a medium of air and water (its pore space). The porosity of a soil sample corresponds to the fraction of its volume not occupied by its constituent particles [13].

The classification of a soil sample is performed by assigning its individual particles to classes according to their size. For example, the United States Department of Agriculture (USDA) defines three soil classes, namely sand, silt, and clay, from the largest to the smallest particles [13]. The relative masses of each component are then compared to determine the texture of a soil sample (*e.g.*, 85% sand-sized particles and 15% silt-sized particles). Within the SPLITS formulation, the dimensions of the particles within each texture class are determined using a particle size distribution provided by Shirazi *et al.* [26].

When running a simulation using the original implementation of the model available for online use [21], henceforth referred simply as SPLITS, one can choose from a fixed set of six soil textures that will guide the size distribution of the particles within each class. These distributions are precomputed using MATLAB [12] scripts, and saved in files that do not change between model runs.

The enhanced model implementation, termed SPLITS-2, incorporates this precomputation as part of the model run framework. This means that the user can input any desired soil texture. The performance overhead is minimal. In fact, the computation of the particle size distribution can be performed fast enough to allow an interactive visualization of the process. We have implemented an utility (Appendix D) that demonstrates this aspect.

2.2 Particle Size Issue

While examining the SPLITS implementation, we noticed an error associated with the generation of the particles. In Listing 1, we provide a code fragment showing how the size of the particles were generated by SPLITS. In this code fragment, the result of the sampling size distribution, `(m_size_warp)(Random::seed1())`, is passed directly to the semi-major axis of the spheroid, denoted by c. However, the result of the sampling size distribution corresponds to the entire major axis, denoted by s in Section 6.2.5 of the original publication describing SPLITS [14], *i.e.*, $s = 2c$. In order to correct this mistake, we simply divided c by 2.

```
Scalar  c = (*m_size_warp)(Random::seed1());
Scalar  a = sphericity * sphericity * c;
return new SpheroidParticle(a, c, Point3::Origin, axis);
```

Listing 1: Code fragment inside `generate()` in `RandomSpheroidParticleGenerator.cpp`.

2.3 Coated Particles Issue

Three types of particles, namely, pure, mixed, and coated, are considered in the model's formulation. A pure particle is made of a single material. A mixed particle is made of two materials combined together using the

Maxwell Garnet equation [13, 14]. Lastly, a coated particle is simulated as a pure particle with a layer (whose thickness is proportional to the particle size [13, 14]) formed by a distinct mineral matrix (possibly embedding impurities like iron oxides) around it.

Here the issue was a variable being passed by reference to different parameters employed by a function used in the simulation of light scattering by a particle coating. One of the parameters was the output direction and the other the input direction. Thus, when the coating light scattering function set the output direction, it unintentionally also set the input direction to this value.

This was a problem because the function checked for an edge case where the light ray should have been reflected, when in fact it was not. However, since the input direction was modified, this edge case was detected more often than it should. We note that the edge case handling procedure set the output ray to a uniform random hemispherical direction. For details, please refer to Appendix A.

While finding this bug was difficult, fixing it was not. All that was required was to make sure that the input and output direction were different variables when calling the function.

We remark that in the deployed version of SPLITS-2 (Section 4), the percentages of mixed, pure and coated particles are to be selected by the user, instead of being limited to a fixed number of choices like in the version of SPLITS available for online use [21].

2.4 Water Saturation Randomization Enhancement

In the model's formulation [13, 14], every time a ray enters the pore space (*e.g.*, after interacting with a particle), the traversing medium (water or air) is stochastically decided based on the water saturation parameter (S between 0 and 1).

This means that for a non-coated particle the pore space is randomized when the ray tries to leave its core. In the case of coated particles, this means that the pore space is randomized when a ray exits the uppermost layer of coating. A concrete example: a ray is about to leave a particle and the soil has 10% water saturation ($S = 0.1$). This means, that each time the ray enters the pore space, there is a 10% chance of it being water and a 90% chance of being air (or vacuum).

In SPLITS-2, we have slightly modified this algorithm. More precisely, the medium changes only when the location of the ray does. The location is relevant because the distances in the coating are assumed to be small, *i.e.*, the ray does not deviate from its entry point to its exit point. This means that, when the ray is inside the coating, the pore medium is fixed as all the points within the coating are considered to be one point on the particle.

Incorporation of Water Film around Particles

The particles of dry sand layers ($S = 0$), albeit immersed in a pore space filled with air, may be encapsulated by water films [18]. In this case, the pore space may have been previously occupied and/or traversed by water, which has either percolated to underneath layers or partially evaporated, leaving only water films around the particles [18]. Related investigations in this area [11, 18] indicate that the thickness of a water film encapsulating a particle is likely to be independent of the particle size.

In the SPLITS-2 implementation, we have incorporated the possibility of having water films around the particles. The value assigned to the film thickness is selected by the user from physically-valid ranges reported in the related literature. This procedure matches the procedures employed in other simulations and analyses involving particulate materials [19, 29]. Moreover, it allows the users to directly control the water film thickness and assess its effects on sand samples subject to varying environmental conditions.

2.5 Clay-Sized Particles Issue

The computation of the mean distance between particles in the online version of SPLITS [21] had an error associated with the presence of clay-sized particles in a given sand sample. This error was prompted by changes in the MATLAB script used to precompute the soil texture option including these particles (Section 2.1). It could result in particles never been hit by traversing rays even though they should be. Examples are provided in Appendix B. This issue has been addressed in SPLITS-2 with the incorporation of the precomputation procedure for the selection of soil texture (Section 2.1), in its correct form for clay-sized particles, to the model run framework. We note that in all of our previous investigations using SPLITS (*e.g.*, [1, 3, 6, 4, 5, 7, 8, 9, 13, 15, 16]), the presence of clay-sized particles was assumed to be negligible, *i.e.*, these particles were not included in the simulations performed during those investigations.

3 Predictability Assessment

In our assessment of the predictive capabilities of SPLITS-2 in comparison with those of SPLITS, we considered samples from four natural sand deposits with distinct morphological and mineralogical characteristics, namely a hematite-rich Australian sand dune, a Saudi Arabian sand dune, a Californian outcrop, and a magnetite-rich Peruvian beach. The actual reflectance curves measured for these samples were made available in the U.S. Army Topographic Engineering Center (TEC) database [25] under the identifications TEC #10019201, TEC #13j9823, TEC #19au9815, and TEC #10039240, respectively.

In the characterization of the selected samples, we considered quartz as their core material and kaolinite as their coating matrix. In addition, we employed mean values for their porosity (0.425), grain roundness (0.482), and grain sphericity (0.798) [8]. The remaining parameter values used in their characterization are given in Tables 1 and 3. In the absence of complete characterization data, the parameter values depicted in these tables were chosen from physically valid ranges provided in the literature [13] so that we could obtain the best possible matches between the model's predictions (obtained using the SPLITS-2 and SPLITS implementations, respectively) and the measured reflectance data.

Note that the percentages of the sand-sized and silt-sized particles depicted in Tables 1 and 3 are employed to compute the dimensions of the samples' grains (whose respective average values are presented in Tables 2 and 4) using a particle size distribution provided by Shirazi *et al.* [26]. Also, based on the samples' descriptions [25], we assumed that the presences of moisture and clay-sized particles were negligible.

Samples	ϑ_{hg}	r_{hg}	ϑ_m	s_a	s_i	μ_p	μ_m	μ_c
Australian dune	0.012	0.8	0.0	90.0	10.0	0.0	50.0	50.0
Saudi Arabian dune	0.012	0.5	0.0	90.0	10.0	0.0	75.0	25.0
Californian outcrop	0.042	0.25	0.0	92.5	7.5	50.0	25.0	25.0
Peruvian beach	0.05	0.375	0.17	95.0	5.0	50.0	0.0	50.0

Table 1: Parameter values used to characterize the four sand samples employed in the comparisons of SPLITS-2 predictions with measured data. The parameter r_{hg} corresponds to the ratio between the mass fraction of hematite to ϑ_{hg} (the total mass fraction of hematite and goethite). The parameter ϑ_m represents the mass fraction of magnetite, which is assumed to appear as pure particles [1]. The texture of the samples is described by the percentages (%) of sand (s_a) and silt (s_i). The particle type distributions considered in the simulations are given in terms of the percentages (%) of pure (μ_p), mixed (μ_m) and coated (μ_c) grains.

Samples	m_a	m_i
Australian dune	0.126	0.022
Saudi Arabian dune	0.126	0.023
Californian outcrop	0.132	0.022
Peruvian beach	0.141	0.021

Table 2: Average dimensions (given in mm) of the major axes m_a and m_i that respectively define the ellipsoids used to represent the sand-sized and the silt-sized particles forming the samples during the simulations performed using SPLITS-2.

Samples	ϑ_{hg}	r_{hg}	ϑ_m	s_a	s_i	μ_p	μ_m	μ_c
Australian dune	0.01	0.75	0.0	85.0	15.0	0.0	100	0.0
Saudi Arabian dune	0.01	0.5	0.0	85.0	15.0	10.0	90.0	0.0
Californian outcrop	0.04	0.0	0.0	85.0	15.0	50.0	0.0	50.0
Peruvian Beach	0.045	0.35	0.17	92.8	7.2	50.0	0.0	50.0

Table 3: Parameter values used to characterize the four sand samples employed in the comparisons of SPLITS predictions with measured data. The parameter r_{hg} corresponds to the ratio between the mass fraction of hematite to ϑ_{hg} (the total mass fraction of hematite and goethite). The parameter ϑ_m represents the mass fraction of magnetite, which is assumed to appear as pure particles [1]. The texture of the samples is described by the percentages (%) of sand (s_a) and silt (s_i). The particle type distributions considered in the simulations are given in terms of the percentages (%) of pure (μ_p), mixed (μ_m) and coated (μ_c) grains.

Samples	m_a	m_i
Australian dune	0.236	0.045
Saudi Arabian dune	0.236	0.045
Californian outcrop	0.236	0.045
Peruvian beach	0.265	0.044

Table 4: Average dimensions (given in mm) of the major axes m_a and m_i that respectively define the ellipsoids used to represent the sand-sized and the silt-sized particles forming the samples during the simulations performed using SPLITS.

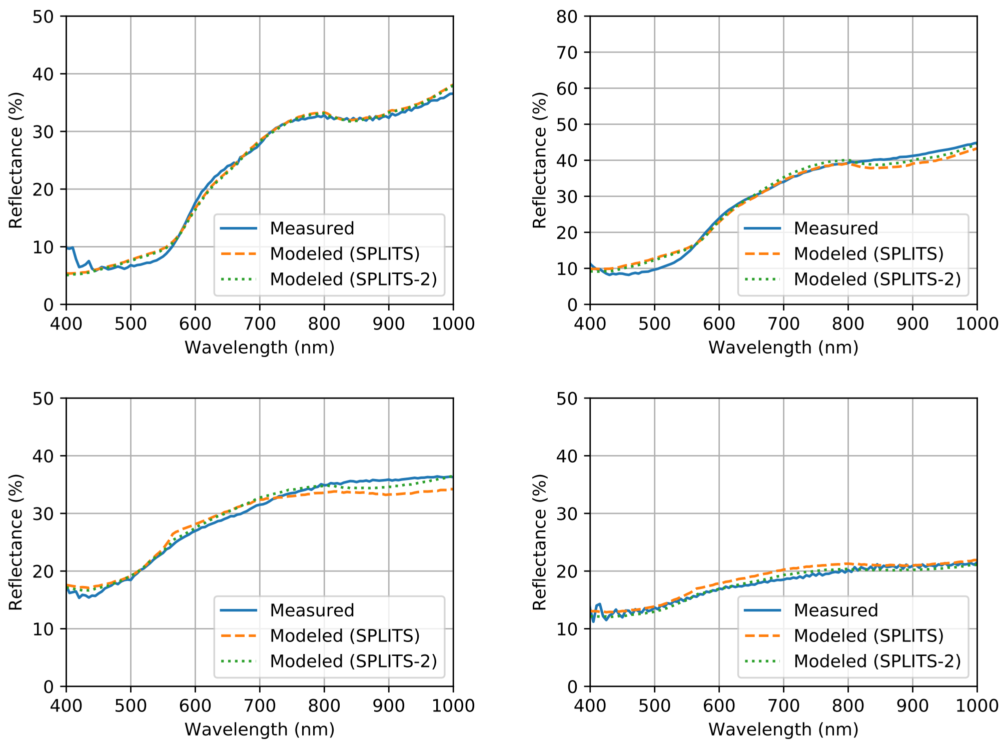


Figure 1: Comparisons of SPLITS-2 and SPLITS predictions with measured data. Top left: Australian dune. Top right: Saudi Arabian dune. Bottom left: Californian outcrop. Bottom right: Peruvian beach. Please refer to Table 1 for the parameter values used by SPLITS-2, and Table 3 for the parameter values used by SPLITS.

For completeness, we also provide root mean square error (RMSE) values computed for the modeled reflectance curves with respect to their measured counterparts. These RMSE values were computed using the following expression:

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\rho_a(\lambda_i) - \rho_b(\lambda_i))^2}, \tag{1}$$

where ρ_a and ρ_b respectively correspond to measured and modeled directional-hemispherical reflectance values, and N is the total number of wavelengths sampled with a 5 nm resolution.

The resulting RMSE values are presented in Table 5. Note that the RMSE values computed for the curves obtained SPLITS-2 were lower than those computer for the curves obtained using SPLITS, indicating closer matches between the former and the measured curves.

Sample	Model Implementation	RMSE
Australian dune	SPLITS	0.0074
	SPLITS-2	0.0068
Saudi Arabian dune	SPLITS	0.0175
	SPLITS-2	0.0132
California outcrop	SPLITS	0.0146
	SPLITS-2	0.0076
Peruvian beach	SPLITS	0.0095
	SPLITS-2	0.0051

Table 5: RMSE values computed for the modeled curves, which were obtained using the SPLITS-2 and SPLITS model implementations, with respect to their measured counterparts [25].

4 SPLITS-2 Online Deployment

The online version of SPLITS-2 [23] is similar to the online version of SPLITS [21], with two main exceptions. Since the soil texture precomputation has been directly integrated into the simulations, it is not restricted to pre-determined distributions.

The other change is that particle type distribution is no longer fixed. Particle type distribution is the choice of how many particles are in each class: pure, mixed, and coated. There is no technical reason in the simulation for this restriction since these values are not used in any precomputation. The reason for this restriction was that the pre-determined distribution values were used by the script that starts the simulation. This restriction was removed in the SPLITS-2 online version.

5 Summary

In this report, we concisely described our work involving the careful examination of the original implementation of the SPLITS model, and introduced SPLITS-2, an enhanced version of that implementation. We have also compared modeled curves, obtained using the SPLITS and SPLITS-2 implementations, with measured ones to demonstrate that the model's predictive capabilities have been preserved by the latter implementation version.

References

[1] G.V.G. Baranoski, B.W. Kimmel T.F. Chen, and E. Miranda. Influence of sand-grain morphology and iron-oxide distribution patterns on the reflectance of sand-textured soils. *IEEE J-STARS*, 7(9):3755–3763, 2014.

[2] G.V.G. Baranoski, T. Dimson, T.F. Chen, B.W. Kimmel, D. Yim, and E. Miranda. Rapid dissemination of light transport models on the web. *IEEE Computer Graphics & Applications*, 32(3):10–15, 2012.

[3] G.V.G. Baranoski and B.W. Kimmel. Can porosity affect the hyperspectral signature of sandy landscapes? In U. Michel and K. Schulz, editors, *Proc. of SPIE, Vol. 10428, Earth Resources and Environmental Remote Sensing/GIS Applications VIII SPIE, Remote Sensing*, pages 104280S1–7, Warsaw, Poland, 2016.

[4] G.V.G. Baranoski, B.W. Kimmel, T.F. Chen, and E. Miranda. Simulating the spectral properties of iron-bearing regions of mars using the SPLITS model. In *2014 IEEE Geoscience and Remote Sensing Symposium*, pages 3013–3016. IEEE, 2014.

[5] G.V.G. Baranoski, B.W. Kimmel, T.F. Chen, and E. Miranda. Assessing the spectral sensitivity of martian terrains to iron oxide variations using the SPLITS model. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 8(7):3404–3413, 2015.

[6] G.V.G. Baranoski, B.W. Kimmel, T.F. Chen, E. Miranda, and D. Yim. Effects of sand grain shape on the spectral signature of sandy landscapes in the visible domain. In *2013 IEEE International Geoscience and Remote Sensing Symposium-IGARSS*, pages 3060–3063. IEEE, 2013.

[7] G.V.G. Baranoski, B.W. Kimmel, and P. Varsa. Assessing the impact of porosity variations on the reflectance and transmittance of natural sands. *Journal of Applied Remote Sensing*, 13(2):024522, 2019.

[8] G.V.G. Baranoski, B.W. Kimmel, P. Varsa, and M. Iwanchyshyn. On the light penetration in natural sands. In *2019 IEEE International Geoscience and Remote Sensing Symposium - IGARSS 2019*, pages 6933–6936. IEEE, 2019.

[9] G.V.G. Baranoski, B.W. Kimmel, P. Varsa, and M. Iwanchyshyn. Porosity effects on red to far-red ratios of light transmitted in natural sands: implications for photoblastic seed germination. In C.M.U. Neale and A. Maltese, editors, *Proc. of SPIE, Vol. 11149, Remote Sensing for Agriculture, Ecosystems, and Hydrology XXI, SPIE, Remote Sensing*, pages 111490O–1–14, Strasbourg, France, 2019.

[10] T.F. Chen, G.V.G. Baranoski, B.W. Kimmel, and E. Miranda. Hyperspectral modeling of skin appearance. *ACM Transactions on Graphics*, 34(3):31, 2015.

[11] J. Czarnecki, B. Radoev, L. Schramm, and R. Slavchev. On the nature of Athabasca oil sands. *Advances in Colloid and Interface Science*, 114:53–60, 2005.

[12] D. Hanselman and B. Littlefield. *Mastering MATLAB 6 A Comprehensive Tutorial and Reference*. Prentice Hall, Upper Saddle River, NJ, 2001.

[13] B. Kimmel and G.V.G. Baranoski. A novel approach for simulating light interaction with particulate materials: application to the modeling of sand spectral properties. *Opt. Express*, 15(15):9755–9777, 2007.

[14] B.W. Kimmel. SPLITS: A spectral light transport model for sand. Master's thesis, D.R. Cheriton School of Computer Science, University of Waterloo, Canada, 2005.

[15] B.W. Kimmel and G.V.G. Baranoski. A compact framework to efficiently represent the reflectance of sand samples. *IEEE Transactions on Geoscience and Remote Sensing*, 47(11):3625–3629, 2009.

[16] B.W. Kimmel and G.V.G. Baranoski. Simulating the appearance of sandy landscapes. *Computers & Graphics*, 34(4):441–448, 2010.

[17] S.R. Van Leeuwen, G.V.G. Baranoski, and Bradley W Kimmel. Revisiting the CLBlood model: Formulation enhancements and online deployment. *Technical Report CS-2017-01, D.R. Cheriton School of Computer Science, University of Waterloo, Canada*, 2017.

[18] A. McCauley, C. Jones, and J. Jacobsen. Basic soil properties. Technical Report Soil & Water, Management Module I, Montana State University, USA, 2005.

[19] A. Mekonen, P. Sharma, and F. Fagerlund. Transport and mobilization of multiwall carbon nanotubes in quartz sand under varying saturation. *Environ. Earth Sci.*, 71:3751–3760, 2014.

[20] Z. Merali. ...ERROR ...why scientific programming does not compute. *Nature*, 467:775–777, 2010.

[21] Natural Phenomena Simulation Group (NPSG). *Run SPLITS Online.* D.R. Cheriton School of Computer Science, University of Waterloo, Ontario, Canada, 2012. http://www.npsg.uwaterloo.ca/models/splits.php.

[22] Natural Phenomena Simulation Group (NPSG). *Sand Data.* D.R. Cheriton School of Computer Science, University of Waterloo, Ontario, Canada, 2012. http://www.npsg.uwaterloo.ca/data/sand.php.

[23] Natural Phenomena Simulation Group (NPSG). *Run SPLITS-2 Online.* D.R. Cheriton School of Computer Science, University of Waterloo, Ontario, Canada, 2020. http://www.npsg.uwaterloo.ca/models/splits2.php.

[24] U.S. Department of Commerce NIST. *NIST/SEMATECH e-Handbook of Statistical Methods*, 2003. https://www.itl.nist.gov/div898/handbook/index.htm.

[25] J.N. Rinker, C.S. Breed, J.F. McCauley, and P.A. Corl. Remote sensing field guide – desert. Technical Report ETL-0588, U.S. Army Topographic Engineering Center, Fort Belvoir, VA, USA, Sept. 1991.

[26] M.A. Shirazi, L. Boersma, and J.W. Hart. A unifying quantitative analysis of soil texture: Improvement of precision and extension of scale. *Soil Sci. Soc. Am. J.*, 52(1):181–190, 1988.

[27] L. Wasserman. *All of statistics: a concise course in statistical inference.* Springer Science & Business Media, 2013.

[28] D. Yim, G.V.G. Baranoski, B.W. Kimmel, T.F. Chen, and E. Miranda. A cell-based light interaction model for human blood. *Computer Graphics Forum*, 31(2):845–854, 2012.

[29] X. Zheng, R. Zhang, and H. Huang. Theoretical modeling of relative humidity on contact electrification of sand particles. *Scientific Reports*, 4:4399, 2014.

A Supplemental Information to the Coated Particle Issue

In this appendix, we provide details about the issue affecting light scattering by coated particles (Section 2.3). The issue was associated with the passing of the same value to two different parameters by reference. The Listing 2 shows some of the code associated with this.

```
if (Random::bernoulli(R))       // ray is reflected
{
    O = Optics::reflect(I, Np);
    assert(equal(O.normSquared(), 1.0));

    if (dot(O, N) * dot(I, N) > 0.0)        // invalid direction
    {
        O = sphericalToCartesian(Random::diffuse(), Basis3::FromW(N));
        assert(equal(O.normSquared(), 1.0));
    }
}
```

Listing 2: Code fragment inside `InterfaceScatter(...)` in `Test1Material.cpp`. Note that O, N and I represent the outgoing, normal and incident vectors, respectively.

The bug occurs in the second `if` statement. The statement is supposed to check whether the outgoing (reflected) ray is propagated to the correct side of the plane as as indicated by the geometric normal N.

In more detail, the symbols that we are interested are the outgoing vector O, the normal vector N, and the incident vector I. Note that if the angle of incidence between O and N is greater than 90°, then the dot product $O \cdot N$ is less than zero. Otherwise, it is greater than zero. Accordingly, the condition $(O \cdot N) * (I \cdot N) > 0$ indicates that the outgoing vector has not changed direction from the incoming vector with respect to the normal. This is an error because the direction relative to the normal must change if we are in the reflected case. The error handling in this case simply sends the ray out on the proper side of the interface with a random hemispherical distribution.

To assess the issue, we must take a closer look at the functions `InterfaceScatter` and `LayerScatter` whose signature is depicted in Listing 3, and how `LayerScatter` is called, which is depicted in Listing 4. In this examination, the arguments of interest are the incident and outgoing ray represented by I and O respectively. When `ParticleScatter(..)` in `Test1Material.cpp` calls `LayerScatter`, it passes the same vector to the incident and outgoing rays. This is so that a second vector to pass by reference does not have to be created. Thus, when `LayerScatter` returns, the new direction is already set, and `ParticleScatter(..)` is ready for the next interaction. However, there is a problem, namely the same vector is passed by reference all the way down to `InterfaceScatter`. This means that in `InterfaceScatter` when `O = Optics::reflect(I, Np);` happens, I is also set to this value. Accordingly, the error checking case that was testing whether $(O \cdot N) * (I \cdot N) > 0$ is now checking if $(O \cdot N) * (O \cdot N) > 0$, which is always true (since: $a * a > 0, \forall a \neq 0$). This is why the error case occurs quite often when the simulation includes light interaction with coated particles.

```
static bool InterfaceScatter(
        const Vector3&                  I,
        const Vector3&                  N,
        const std::complex<Scalar>&     m1,
        const std::complex<Scalar>&     m2,
        Scalar                          roundness,
        Vector3&                        O)

static int LayerScatter(
        const Vector3&                  I,
        const Vector3&                  N,
        const std::complex<Scalar>&     m1,
        const std::complex<Scalar>&     m2,
        Scalar                          alpha,
        Scalar                          thickness,
        Scalar                          roundness,
        Vector3&                        O
)
```

Listing 3: The signatures of `InterfaceScatter` and `LayerScatter` functions in `Test1Material.cpp`.

```
850 int dlayer = LayerScatter(O.direction(), N, m1, m2, alpha,  pd.coatings[layer].thickness *↩
        particle.diameter(), roundness[layer], O.direction());
```

Listing 4: Code fragment showing `LayerScatter` being called by `ParticleScatter` in `Test1Material.cpp`. Note that `O.direction()` is passed to `LayerScatter` (Listing 3) as both I and O parameters.

We note that this issue does not affect non-coated particles as `CoreScatter` does not use Interface scatter. `CoreScatter` passes the arguments differently so the two are not references of the same vector.

B A Worked Example of the Issued Involving Clay-Sized Particles

In this appendix, using a worked example, we examine why the precomputed soil textures that included clay-sized particles led to problems (Section 2.5), and the effects of these problems on the simulations. This example considers a sand sample characterized by the default parameters values employed in the online version of SPLITS [21] as well as a soil texture comprising 90% sand-sized and 10% clay-sized particles. Pure and coated particles are not considered, only mixed ones (Section 2.3). This results in four particle type generators (Appendix C) depicted in Table 6.

The (custom-defined) `mean_distance` (Appendix C.2) represents the average distance between particles of a given type. To derive the `mean_distance` values shown in Table 6, we present additional information. In the selected sample, r_{hg} (the ratio between the mass fraction of hematite to the total mass fraction of hematite and goethite) is equal to 0.75. In addition, the sample is characterized by a porosity equal to 0.425. Thus, when we derive the `mean_distance` value for particle obtained using the particle type 1 (Table 6), we know that this particle is sand-sized, has hematite, and it is within the portion of the sample occupied by particles (because it is a particle!). This means that, for the particle type 1, the (custom-defined) `concentration` quantity is computed

particle type	soil class	iron oxide	*mean_distance* (*m*)	*specific_extinction* (m^{-1})	*concentration*
1	sand	hematite	0.3988	6.4756	0.3881
2	sand	goethite	1.1926	6.4756	0.1295
3	clay	hematite	0.0038353	6047.76	0.0431
4	clay	goethite	0.011493	6047.76	0.0144

Table 6: Values used by four particle type generators within the SPLITS implementation considering a sample with 10% clay-sized and 90% sand-sized particles. Soil class describes the USDA soil classification (sand, silt, or clay) according to the grain size. Iron oxide describes indicates which iron oxide in considered in a given particle type. The distribution of iron oxides is affected by the r_{hg} parameter (the ratio between the mass fraction of hematite to the total mass fraction of hematite and goethite). The quantity *mean_distance* corresponds to the average distance in meters between particles generated by a given particle type generator. The quantity *specific_extinction* has units $\frac{m^2}{m^3}$, that is square meters of particle cross section per cubic meter of particle volume. The quantity *concentration* corresponds to the fraction of the selected soil sample occupied by particles (containing a specific iron oxide) associated with a given particle type generator, and it is dimensionless.

as:

$$concentration = 0.90 * 0.75 * (1 - 0.425) = 0.3881. \tag{2}$$

We note that this quantity also takes into account the particle type, which is omitted in Equation 2 because we are considering 100% mixed particles in this example. Moreover, it also worth mentioning that the sum of concentrations of all particle types composing a given sample is equal to $1 - porosity$.

The (custom-defined) `specific_extinction` quantity corresponds to the cross-sectional area to volume ratio of a given particle type. It depends only on the shape and size of the particle. Finally, the `mean_distance` is computed as:

$$mean\_distance = \frac{1}{specific\_extinction * concentration}. \tag{3}$$

Listing 5 shows how this is computed in the SPLITS implementation.

```
Scalar   extinction = m_particles.back().generator->extinction();
m_particles.back().meanDistance = 1.0 / (extinction * m_particles.back().concentration);
```

Listing 5: How mean distance is computed in `Test1Material.cpp`.

The `mean_distance` computed using Equation 3 is then used to sample the `distance` to the next particle generated by a specific generator by sampling an exponential distribution:

$$\text{distance} = mean\_distance * -ln(rand()), \tag{4}$$

where $rand()$ is a function that returns a uniform random number in $[0, 1]$. Listing 6 shows how and where this is called in the code.

```
Scalar   next_d = Random::exponential(m_particles[i].meanDistance);
```

Listing 6: How distance to next particle is computed in `Test1Material.cpp`.

In closing, as indicated by the values presented in Table 6, due to the error in the precomputation employed by SPLITS, the distance between two sand-sized particles are 0.39 *m* and 1.19 *m* respectively. Thus, in practice, a ray would not interact with these particles. We remark that in all of our previous investigations using SPLITS, the presence of clay-sized particles was assumed to be negligible, *i.e.*, these particles were not included in the simulations employed in those investigations.

particle type	soil class	iron oxide	*mean_distance* (m)	*specific_extinction* (m^{-1})	*concentration*
1	sand	hematite	1.507E-4	17105	0.3881
2	sand	goethite	4.515E-4	17105	0.1295
3	clay	hematite	8.31E-6	2790805	0.0431
4	clay	goethite	2.50E-5	2790805	0.0144

Table 7: Values used by four particle type generators within the SPLITS-2 implementation considering a sample with 10% clay-sized and 90% sand-sized particles. Soil class describes the USDA soil classification (sand, silt, or clay) according to the grain size. Iron oxide describes indicates which iron oxide in considered in a given particle type. The distribution of iron oxides is affected by the r_{hg} parameter (the ratio between the mass fraction of hematite to the total mass fraction of hematite and goethite). The quantity *mean_distance* corresponds to the average distance in meters between particles generated by a given particle type generator. The quantity *specific_extinction* has units $\frac{m^2}{m^3}$, that is square meters of particle cross section per cubic meter of particle volume. The quantity *concentration* corresponds to the fraction of the selected soil sample occupied by particles (containing a specific iron oxide) associated with a given particle type generator, and it is dimensionless.

Table 7 shows the distance values obtained using SPLITS-2. Note how, even for the largest particles, the mean distance between particles is under 1 mm.

C Selected Details of Inner Workings of SPLITS

Within the stochastic formulation [13, 14] employed by SPLITS and SPLITS-2, particles are generated on the fly a light ray traverses a sand sample. This requires two things: a set of particle generators and a set of mean distances. For each desired particle type in a simulated sample, there is one particle generator and one mean distance. The specifics of a particle generator depend on the composition and geometry of the particle type to be generated. For example, as shown in Appendix B, there can be one generator that makes sand-sized hematite-containing particles. The details regarding how the mean distances are computed are presented in the next sub-section.

In the original SPLITS implementation, the two most important files are `runspectro` and `Test1Material.cpp`. The file `runspectro` is a bash script that setups up the simulation and calculates material (such as hematite) concentrations from the input parameters. This procedure is described in Sections 6.2.4 and 6.2.5 of the original publication [14] describing the model. In the SPLITS-2 implementation, this is done by `SplitsSoil.java`.

The class representing the sample to be interacted with is `Test1Material.cpp`. It has all of the particle generators and simulates how light interacts with the sample. It does this by choosing the next particle type to generate, calling one of the generators, and propagating a light ray through the particle. In the SPLITS-2 implementation, this work is done by `ParticleSurfaceScatterer.java`.

C.1 Warping functions

A warping function is a function that takes a uniform random number in [0, 1], and returns a number with some defined distribution. This is usually called the quantile function, and it is the inverse of the cumulative distribution function (CDF) [27]. Some authors call it the Percent Point Function [24]. Let's consider a probability density function (PDF) $f(x)$, with a range from x_{min} to x_{max}. We know that:

$$\int_{x_{min}}^{x_{max}} f(x)\,dx = 1. \tag{5}$$

We also know that the CDF, denoted by F, is given by:

$$F(x) = \int_{x_{min}}^{x} f(u)\,du, \tag{6}$$

with $F(x)$ only being defined for $x \in [x_{min}, x_{max}]$. This means that the CDF is also the probability that another number taken from the distribution will be lower than x. This can also be thought of as the percentage of all the members of $[x_{min}, x_{max}]$ below x.

This is where the quantile function, $Q(p)$, inverts the CDF so that the input is now a percentage, or probability, and the output is a member of $[x_{min}, x_{max}]$:

$$Q(p) = \{x \in {}_{[x_{min},x_{max}]} \ : \ p = F(x)\}, \tag{7}$$

or more simply:

$$Q = F^{-1}. \tag{8}$$

Now we can sample the distribution given by f by simply passing a random number in $[0, 1]$ to Q. For example, let us choose $p = 0.9$. This means that the 90th percentile value is returned. We know that there is a 10% chance of getting a value higher, and from our uniform distribution on $[0, 1]$, we can clearly see that p has a 10% chance of being higher than 0.9. This is the basis of how the warping functions generate probability distributions within the model's formulation, *i.e.*, they 'warp' the 0 to 1 range of the input to the desired distribution in the output.

C.2 Computation of Mean Distance

In the SPLITS implementation, the computation of the `mean_distance` between two particles of the same type is spread over 5 files:

- `Test1Material.cpp`: $mean\_distance = 1/(extinction * concentration)$
- `RandomSpheroidParticleGenerator.cpp`: $extinction = specific\_extinction$
- `random_spheroid.tcl`: $specific\_extinction = extinction * extinction\_factor$. (These values for these two parameters come from particle size distribution (`psd`) and sphericity (`shirazi`) files).
- `t1m.tcl`: $concentration = particle\_concentration = concentration * fraction$, where $fraction$ is calculated as $fraction = particle\_fraction_j/total\_particle\_fraction$. The variable $particle\_fraction_j$ passed by `runspectro` and it represents the fraction of the particle volume occupied by the particle type j.
- `runspectro`: $concentration = (1 - porosity)$. This $concentration$ value gets stored in a variable internally termed f0. Note that the calculation of $particle\_fraction_j$ from input parameters is performed within this file as well (Section B).

The precomputed warping functions that `random_spheroid.tcl` uses are stored under `particles/warp/` in the SPLITS implementation. An example of a soil texture warping file is: `shirazi_0_10_90_sand.tcl`. This refers to the soil portion of a soil texture comprising 90% sand-sized particles and 10% silt-sized particles. The sphericity warping functions are stored under `particles/warp/tmp`, and are generated for each run. This allows custom values for sphericity to be set. To generate warping function files see `mkpsdtcl.m` (MaKe Particle Size Distribution TCL file) and `mksphtcl.m` (MaKe SPHericity TCL file). These two files (`mkpsdtcl.m` and `mksphtcl.m`) are the ones associated with the issue discussed in Section 2.5.

In the SPLITS-2 implementation, the quantity `mean_distance` is calculated by function, namely `RandomlyOrientedParticleParameter.java`, that takes soil texture and particle sphericity distribution as arguments.

C.3 Particle Shape and Size

In this section, we provide details on how the size of the particles are chosen. This section can be seen as an extension of the presentation provided in Section 2.2. Please refer to Listing 7 for how the items discussed here are actually used, and Appendix C.1 for the basics of how warping functions work. This section is also meant as a companion to the material presented in Section 6.3.3 of the original publication [14] describing the model.

We start at the particle generator where these distributions are used. Note that in the SPLITS implementation these files were located under `mist/particles/warp/`. The `RandomSpheroidParticleGenerator` takes two of these warping functions: one for size and one for sphericity. Here sphericity means the Riley sphericity [13], denoted by Ψ, which is given by:

$$\Psi = \sqrt{\frac{a}{c}}, \tag{9}$$

in the SPLITS implementation. We also have:

$$a = \Psi^2 * c, \tag{10}$$

where a and c represent the major and minor axes of a prolate spheroid (where $a > c$) representing a particle [13, 14].

Listing 7 shows how a particle generator calculates the size and shape of a particle. Note how first the size distribution is sampled first, then the sphericity.

```cpp
IParticle* RandomSpheroidParticleGenerator::generate() const
{
    Vector3 axis = sphericalToCartesian(Random::uniformOnSphere());
    Scalar  sphericity = (*m_sphericity_warp)(Random::seed1());

    assert(inRangeOC(sphericity, 0.0, 1.0));

    Scalar  c = (*m_size_warp)(Random::seed1());
    Scalar  a = sphericity * sphericity * c;
    return new SpheroidParticle(a, c, Point3::Origin, axis);
}
```

Listing 7: The function used to generate the particles. The function `Random::seed1()` generates a random number between 0 and 1.

The warping function `m_size_warp` is a discretization of the following continuous probability distribution function (PDF):

$$f_s(x) = \frac{1}{C_1} f_g(x) x^{-2}. \tag{11}$$

Equation 11 corresponds to Equation 6.23 provided in the original publication [14] describing the model. The parameter f_S is the probability that a particle with size x is hit by a ray. Here $f_g(x)$ is the log-normal probability density function associated with the Shirazi soil texture [13, 14], *i.e.*, the fraction of volume of the sample that is size x. C_1 is a constant of integration such that the probability distribution f_S integrates to 1. Initially, it looks as if the probability of hitting a particle should go up as the square of the size, not down. However, the probability does go down because the smaller particles have a greater surface area to volume ratio. Thus, when occupying the same volume, a set of smaller particles has greater cross sectional area, and it is more likely to be hit by a traversing ray.

The warping function `m_sphericity_warp` is slightly different because the distribution of Ψ is defined over the entire sample. We know that the mean and standard deviation of the sphericity are: $\overline{\Psi}$ and σ_Ψ. Hence, the PDF that `m_sphericity_warp` is following is $f_\Psi(y)$, *i.e.*, the probability that a ray hits a particle with a sphericity y. The complication arises because changing the sphericity changes the surface area to volume ratio. In Equation 6.12 provided in the original publication [14] describing the model, the parameter $A_V(\Psi)$ corresponds to the surface area to volume ratio of a unit sized prolate spheroid.

Finally, let us examine Equation 6.24 provided in the original publication [14] describing the model. The function $f_\Psi(y)$ corresponds to the PDF of a particle with sphericity y being hit by a ray as it travels through a sample:

$$f_\Psi(y) = \frac{1}{C_2} A_V(y)\, \Phi(\overline{\Psi}, \sigma_\Psi^2)(y), \tag{12}$$

where $\Phi(\overline{\Psi}, \sigma_\Psi^2)(y)$ is the PDF of a normal distribution with mean $\overline{\Psi}$, variance σ_Ψ^2, and it is evaluated at y. The function $f_\Psi(y)$ corresponds to a normal distribution scaled so that sphericities with higher surface to volume ratios are more common. Once again, C_2 is a constant chosen so that the probability density integrates to 1.

To actually invert the PDF's into warping functions, numerical methods are used in precomputation steps, which are implemented using MATLAB.

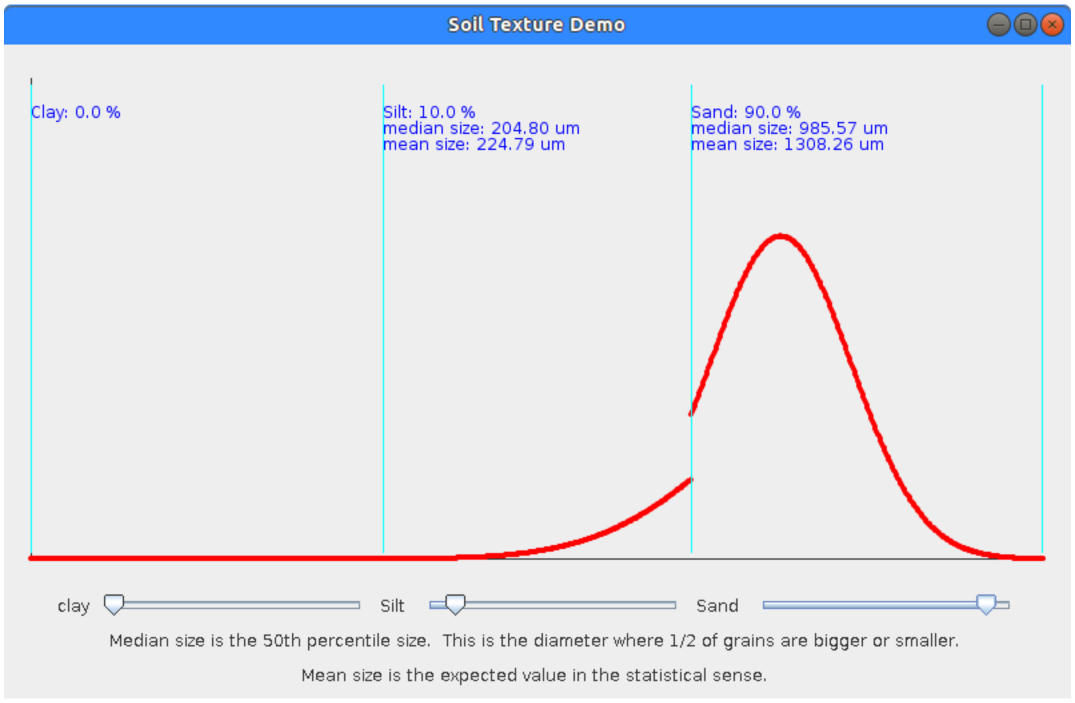
D Visualization of Distinct Particle Size Distributions

We have developed an utility (for internal use) to enable the visualization of intermediate simulation steps in SPLITS-2. A screenshot of the interactive user interface is shown in Figure 2. To use it, one simply adjusts the sliders at the bottom labeled: "Clay", "Silt", and "Sand". The graphs display the probability density as a function of grain size for each soil separate. Each light blue bar from left to right is a grain size of: 0.05, 2, 50 and 2000 μm. These are chosen because they define the limits between sand, silt, and clay in the soil texture computation.

Since the graphs correspond probability distributions, they integrate to 1. However, since the x-axis is not evenly spaced (it is logarithmically scaled), the area under this graph is not a constant. The use of an exponential scaling on the x-axis was chosen because each soil separate has a log-normal distribution. Hence, when plotted considering a logarithmically scaled axis, the probability distribution of size looks like a normal distribution. The y-axis is (linearly) scaled so that most of the graph is visible for most choice of inputs.

The median and mean size in each separate correspond to the soil distribution in that size class. The median size means that half of all particles in that soil separate will be larger and half smaller (the 50th percentile). The mean size is the expected value of the distribution.

To run the utility, one needs to call `SoilTexture.SoilDemo();` in the `main()` function of `testing/Main.java`, and then set the main class of the model to `testing/Main.java` so that the `main()` function gets called. The main class is a configuration in `pom.xml` called `<mainClass>`. This field tells the build system in which file the execution beings.



Median size is the 50th percentile size. This is the diameter where 1/2 of grains are bigger or smaller.

Mean size is the expected value in the statistical sense.

Figure 2: Interactive interface showing the probability density function of various particle sizes. The x-axis corresponds to particle size. It increases from left to right, and it is logarithmically scaled.