Minimal Packing Sets

Lorne Arnold
University of Washington Tacoma
arnoldl@uw.edu

Abstract

Soil is fundamentally a discrete material that is, nevertheless, commonly modeled as a continuum because of the computational expense of large-scale discrete element models (DEMs). Even at lab specimen scales, DEM's computational cost may be substantial depending on the grain sizes being modeled. Despite these limitations, discrete models have proven useful in furthering our understanding of soil mechanics because they can spontaneously replicate realistic soil behavior as an emergent macro-scale property from a collection of particles following relatively simple interaction rules. This makes DEM an attractive tool for a multi-scale modeling approach where the constitutive behavior of a representative volume element (RVE) is characterized with a discrete model and applied at a larger scale through a continuum model. The ability of the RVE to represent a soil depends strongly on an appropriate grain size distribution match. However, in order to achieve computationally feasible models, even at lab scales, DEM simulations often use larger minimum particle sizes and more uniform distributions than their intended targets. Intuitively, discrete matches of different grain size distributions (GSDs) will require vastly different numbers of particles. But the precise relationship between GSD characteristics and the number of particles needed to match the distribution (and by extension the associated computational cost associated) is not intuitive. In this paper, we present the minimal packing set (MPS) concept. The minimal packing set is the smallest set of discrete particles needed to match a given GSD. We present a method for determining the MPS for any GSD and discuss strategies for finding the smallest MPS within a set of tolerances on the GSD. When mapped onto USCS classifications, A mapping of MPS onto USCS classification reveals a highly non-linear relationship between soil classification and computational cost of DEM modeling across several orders of magnitude.

Introduction

```
import sys
sys.path.append('..')
import gsd_lib as gl
import numpy as np
import scipy.stats as stats
import matplotlib.pyplot as plt
```

Intro text.

Soil is discrete. Several frameworks use discrete approaches, but they're difficult to manage and expensive to run.

But mechanics is mechanics and you can't model something well that doesn't replicate its underlying physical behavior. If you can replicate the underlying mechanisms, the macro-scale behavior should emerge.

The RVE concept is somewhat tied to grain size distribution (but not completely).

What is a grain-size distribution? How can it be described mathematically? How precise does a GSD need to be and how would one even measure that?

Which features of a GSD contribute to more or less computationally intense models?

Mathematical GSD

To answer these questions, we need to create rigorous mathematical definitions for soil samples, sieve stacks, and a grain size distribution. Conceptually, each of these are collections of items (i.e., sets) with specific restrictions.

Samples are collections of particles. For the purposes of this paper, assume that particles are spherical.

Sample S:

$$S = \{(x_i, q_i) : i \in I_S\}$$

where:

$X_S = \{x_i : i \in I_S\}$ with $x_1 < x_2 < \cdots < x_{n_S}$ (ordered sizes)

$Q = \{q_i : i \in I_S\}$ (corresponding quantities)

$X_S \subset \mathbb{R}^+$ and $Q \subset \mathbb{Z}^+$

Grain Size Distribution G:

$$G = \{(x_j, v_j) : j \in I_G\}$$

where:

$X_G = \{x_j : j \in I_G\}$ with $x_1 < x_2 < \cdots < x_{n_G}$ (ordered sizes)

$V = \{v_j : j \in I_G\}$ (corresponding volumes)

$X_G \subset \mathbb{R}^+$ and $V \subset \mathbb{R}^+$

Completeness Condition for G: A grain size distribution G is **complete** if and only if:

$$v_{n_G} = 0$$

Updated Comparison Conditions (G describing S):

Condition 1: G describes S if $\min(X_G) < \min(X_S)$ and $\max(X_G) \geq \max(X_S)$

Condition 2: G describes S efficiently if $\forall(x_j, x_{j+1})$ consecutive in X_G, $\exists x_i \in X_S$ such that $x_j < x_i \leq x_{j+1}$

Condition 3: G describes S articulately if $\forall(x_i, x_{i+1})$ consecutive in X_S, $\exists x_j \in X_G$ such that $x_i \leq x_j < x_{i+1}$

Condition 4: G describes S accurately if for all consecutive pairs (x_j, x_{j+1}) in X_G:

$$v_j = \sum_{\substack{i \in I_S \\ x_j < x_i \leq x_{j+1}}} q_i \cdot f(x_i)$$

where $f(x)$ is a scaling function that converts size to volume.

Physical meaning

Practically, we use masses, rather than volumes in grain size distributions. Assuming the particles are of the same density, we can add a mass component to G by multiplying V by the density, ρ.

For a complete grain size distribution, we can also define a percentage (most often by mass), Λ, for each size.

The conditions above are focused on how G describes S, but in reality, we develop grain size distributions from direct measurement of samples. The question at hand is how to generate a sample that is a good match to a known grain size distribution. We can say that S matches G if G accurately describes S.

Note that conditions 1, 2, and 4 are often met in laboratory testing, while condition 3 is practically impossible to meet in the lab, but easily met in discrete element simulations.

Solving for the minimal packing set

Let's start with the simple case where G describes S efficiently and articulately (i.e., there is one and only one particle size retained on each sieve).

Uniform particle sizes in each sieve

Select a single particle size to represent all possible particle sizes between two sieve sizes (i.e., desc (G, S) is articulate).

Representative radius

Imagine a bin with a lower limit of radius, r_{low} and an upper limit of radius r_{high}. The distribution of the number of particle radii within the bin is given by a probability density function, $P(r)$. Selecting a "representative" value for the bin, r_{rep} would likely involve some measure of the average particle size, either by radius or volume. A representation based on volume is more appealing since the overall GSD is volume-weighted.

In absence of any other reason to suspect a specific distribution, a uniform distribution seems like a reasonable choice. Under a uniform distribution of radii, the mean radius would be the average of r_{low} and r_{high}. The radius at the mean volume of this same distribution would given by:

$r_{rep} = r_{low} + \frac{r_{high} - r_{low}}{p}$

where p is between $\sqrt[3]{4}$ and 2 and $\frac{r_{low}}{r_{high}}$ is between 0 and 1.

These come from the range of possible differences between the radius associated with the mean volume and the mean radius. If $\frac{r_{low}}{r_{high}}$ is zero, there is no lower limit to the range (i.e., r_{low} is infinitely small), $r_{mean} = r_{high}/2$, and the mean volume will be $\frac{\pi r_{high}^3}{3}$. The radius associated with the mean volume in this case is $\frac{r_{high}}{\sqrt[3]{4}}$. This corresponds to $p = \sqrt[3]{4}$.

As $\frac{r_{low}}{r_{high}}$ *approaches* one, the difference between r_{low} and r_{high} becomes very small and the radius associated with the mean volume approaches the mean radius. This corresponds to $p = 2$.

If $\frac{r_{low}}{r_{high}}$ is one, there is no difference between r_{low} and r_{high} (i.e., there is no variation in radii), and the mean volume will be $\frac{4\pi r_{high}^3}{3}$. The radius associated with the mean volume in this case is simply r_{high}. This corresponds to an indeterminate condition for p.

Therefore, we can use the following equation to select an appropriate value of p to describe a volume-based representative radius for a uniform distribution of particles between r_{low} and r_{high}:

$$p\left(\frac{r_{low}}{r_{high}}\right) = \left(2 - \sqrt[3]{4}\right)\frac{r_{low}}{r_{high}} + \sqrt[3]{4}$$

with domain $D : [0, 1]$ and range $R : \left[\sqrt[3]{4}, 2\right]$.

To confirm, create a plot of 1000 uniform distributions with $\frac{r_{low}}{r_{high}}$ between 0 and 1 with 1000 particles each and plot the resultant value of p along with the analytical value.

```
# Create a function that calculates p assuming that r_rep is the radius of a sphere with the same volume as the mean volume of the spheres in the distribution
def p_r_rep(r_low, r_high, size=1000, verbose=False):
    """
    Calculate the parameter 'p' from the radius of the mean volume of spheres with random uniformly distributed radii.
    """
    r = stats.uniform.rvs(loc=r_low, scale=r_high-r_low,size = size)
    v = 4/3*np.pi*r**3
    r_mean = np.mean(r)
    v_mean = np.mean(v)
    r_v_mean = (3*v_mean/(4*np.pi))**(1/3)
    if verbose:
        print(
            f"""For a uniform random distribution of radii from {r_low} to {r_high} with {size} samples, \nthe mean radius is {r_mean:.2f}, \nthe mean volume is {v_mean:.2f}, \nand the radius calculated from the mean volume is {r_v_mean:.2f}""")
    return (r_high-r_low)/(r_v_mean-r_low)

def p_analytical(r_low, r_high):
    return (2-4**(1/3))*(r_low/r_high) + 4**(1/3)

r_high = 10
```

```
size = 1000
r_ratio = np.linspace(0,0.999,size)
p = []
for r in r_ratio:
    p.append(p_r_rep(r_high*r, r_high))
plt.scatter(r_ratio, p, marker="o", alpha=0.5)
plt.plot([r_ratio[0],r_ratio[-1]],            [p_analytical(r_high*r_ratio[0],
r_high),p_analytical(r_high*r_ratio[-1], r_high)], linestyle="--", color="k")
plt.xlabel("$r_{low}/r_{high}$")
plt.ylabel("$p$")
text_plot = plt.text(0.0, 2.0,
f"""Values for $p$ in $r_{{rep}} = r_{{low}} + \\frac{{r_{{high}}-r_{{low}}}}
{{p}}$
based on uniform random distributions with {size}
samples of radii from $r_{{low}}$ to $r_{{high}}$""")

text_plot = plt.text(0.6, 1.65,
"""Analytical solution (dashed)
$p = (2-\\sqrt[3]{{4}}) \\frac{{r_{{low}}}}{{r_{{high}}}} + \\sqrt[3]{{4}}$
""")
plt.show()
```

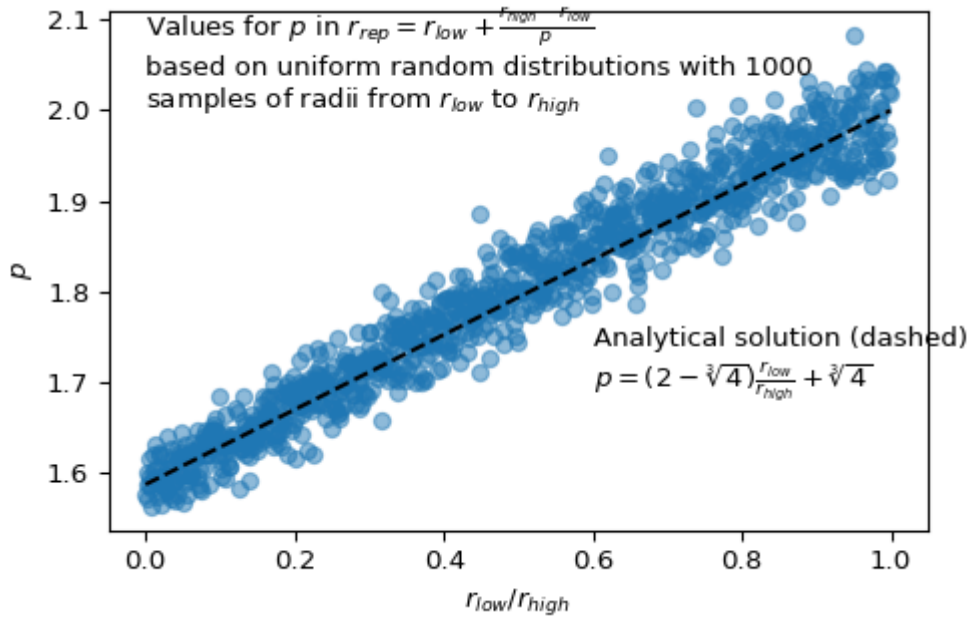

Figure 1: Representative radius for a uniform distribution of particle sizes.

Particle quantities

Given a grain size distribution G of spherical particles, the minimal packing set, S_{mp} is the smallest sample (i.e., minimizing $\| S \|$) that provides a sufficiently good match to G. The number of sizes in S will be $n_G - 1$.

In order to match the G, we need to fully capture the relations between the particle sizes in terms of ratios of: total mass, individual particle mass, and number of particles. These ratios can be described as:

$$\Phi = \left\{ \frac{m_j}{m_{n_G}} : j \in I_G \right\} \text{ ; with elements } \phi_i$$

$$\Xi = \left\{ \frac{x_j^3}{x_{n_G}^3} : j \in I_G \right\} \text{ ; with elements } \xi_i$$

$$Y = \left\{ \frac{\phi_j}{\xi_{n_G}} : j \in I_G \right\} \text{ ; with elements } \upsilon_i$$

If, by some happy accident, Y can be expressed as a fraction of integers, the quantity of particles of each size in S would be described by

$$Q = Y \cdot \text{lcm} \left\{ d_i : \frac{n_i}{d_i} \in Y \right\}$$

Unfortunately, it cannot be assumed that Y can be expressed as a fraction of integers. Nevertheless, we can use multiples of Y to find the closest discrete sample possible with any non-zero number of largest particles, q_{nS}. And for each approximation, the error between the target G and sample, S_i can be calculated. If some practical tolerance for error is known, incremental increases in q_{nS} can be applied to generate samples until the resulting error is less than the tolerance.

In the event that Y can indeed be represented as a fraction of integers, q_{nS} will be equal to $\text{lcm} \left\{ d_i : \frac{n_i}{d_i} \in Y \right\}$.

Insert plot of error v. n ### Distributed particle sizes in each sieve The size of the sample can be minimized further if the $\text{desc}_{\text{art}}(G, S) = \text{true}$ requirement is dropped. Under these conditions, an arbitrary number of sizes between the sizes in G. This represents a more realistic model of laboratory test results. But we don't need to identify all the sizes. In fact, for any distribution of sizes in a range, we can find a single size that can represent the same mass with the same number of particles.

We need to bound the possible values of Ξ and Y with the minimum ($-$) and maximum ($+$) allowable sizes in all but the largest bin. We can then repeat the incremental procedure described in the previous section until the ranges in $Y_- : Y_+$ span integer values. Selecting the lowest spanned integers in this range produces an integer representation, Y_{INT}, that will be possible to create using allowable sizes, which will become the Q for our minimal packing set sample. Then, using functions for Ξ, Φ, and the appropriate scaling functions, we can solve for the sizes, X, that will satisfy the match between G and S when $S = \{(X, G)\}$.

$$P = \sqrt[3]{\Phi}$$

$$X = P \cdot x_{n,S}$$

Note that this solution for the minimal packing set is dependent on the selected value for $x_{n,S}$. Because the maximum particle size, $x_{n,S}$ controls the packing set, and a smaller value for $x_{n,S}$ will tend to minimize the set, if the full range of possible $x_{n,S}$ is considered, $x_{n,S}$ and $x_{n-1,S}$ will tend to converge. The result of that convergence would be equivalent to the solution for a target G with $x_{n-1,S} \approxeq x_{n,S} = x_{n-1,G}$ and the $x_{n-1,G}$ sieve removed. However, it may not always be the case that $x_{n,S}$ converges to $x_{n-1,G}$. By testing different samples with size ratios P and x_{nG} between its min and max value, we can find the value of x_{nG} that creates a sample that is accurately and efficiently described by G. A plot of $\| S_{mp} \| (p)$ would help identify whether any optimal solutions exist other than $x_{n-1,S} \approxeq x_{n,S} = x_{n-1,G}$. This probably depends on the target G.

Regarding $\| S_{mp} \|$, it tends to be controlled by two factors: the maximum value in Y and the value in the last entry in Y. In general terms, the first is a function of the ratio of the largest to smallest particles in the sample. The second is a function of the curvature of the sample.

```
from gsd_lib import GSD, MinimalPackingGenerator
import numpy as np
import matplotlib.pyplot as plt

d_max_d_min = []
m_max_m_min = []
norm_s = []
q_ns = []
errorses = []
iterations = []

sieves = 15
rng = np.random.default_rng()

for i in range(10000):
    # Example usage similar to the original main function

    # d = unique_random_sorted(sieves, 0.075, 75)
    d = np.array([0.00075,0.075, 0.15, 0.3, 0.6, 1.18, 2.36, 4.75, 9.5, 19, 25,
37.5, 50, 63, 75])

    mass = (101 - 1) * rng.random(sieves) + 1
    mass[-1] = 0.0  # Ensure last mass is zero
    # mass = [6.93546639, 12.49031735, 9.29866496, 8.61983351, 0.0]

    # Create GSD
    g = GSD(sizes=d, masses=mass)

    # Create minimal packing generator
    sgen = MinimalPackingGenerator(g)
```

```
    s = sgen.mps

    d_max_d_min.append(s.sizes[-1] / s.sizes[0])
    m_max_m_min.append(s.total_masses[-1] / s.total_masses[0])
    norm_s.append(np.sum(s.quantities))
    q_ns.append(s.quantities[-1])
    errorses.append(sgen.errors)
    iterations.append(sgen._iteration)

d_max_d_min = np.array(d_max_d_min)
m_max_m_min = np.array(m_max_m_min)
norm_s = np.array(norm_s)
q_ns = np.array(q_ns)
# errorses = np.array(errorses)
iterations = np.array(iterations)
```

```
fig, ax = plt.subplots()

sc = ax.scatter(1/m_max_m_min, norm_s, c=q_ns, s=q_ns, cmap="viridis",
alpha=0.6)
# sc = ax.scatter(m_max_m_min, iterations, c=q_ns, s=q_ns, cmap="viridis",
alpha=0.6)
ax.set_xscale('log')
# ax.set_yscale('log')
```

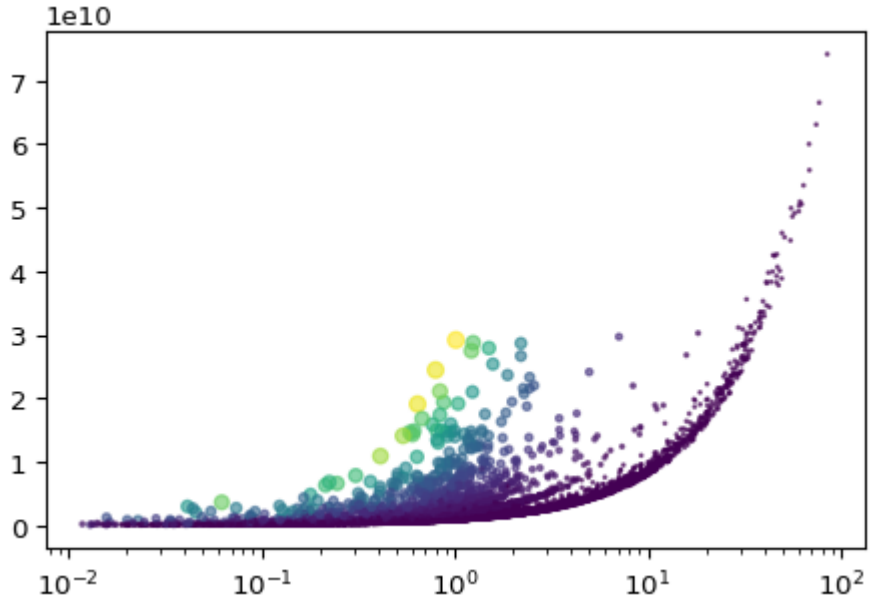

Figure 2: Exponential increase in N with min/max ratio

Implications for DEM modeling

The algorithms described above can 1. Find a first-order approximation of the minimal packing. 2. Quantify the error in approximate minimal packings. 3. Find a rigorous solution minimal packing.

```
fig, ax = plt.subplots()

for i, error in enumerate(errorses):
    # print(error)
    # print(i)
    try:
       # ax.scatter(np.linspace(1, iterations[i], iterations[i]), np.max(error,
axis=1), alpha=0.05, c="k")
        ax.plot(
            np.linspace(1, iterations[i], iterations[i]),
            np.max(error, axis=1),
            alpha=0.1,
        )
    finally:
        continue
# sc = ax.scatter(d_max_d_min, norm_s, c=q_ns, s=q_ns, cmap="viridis", alpha=0.6)
# sc = ax.scatter(iterations, q_ns, c=q_ns, cmap="viridis", alpha=0.6)
ax.set_xscale("log")
ax.set_yscale('log')
# ax.set_ylim(1e-6, 1e-1)
```

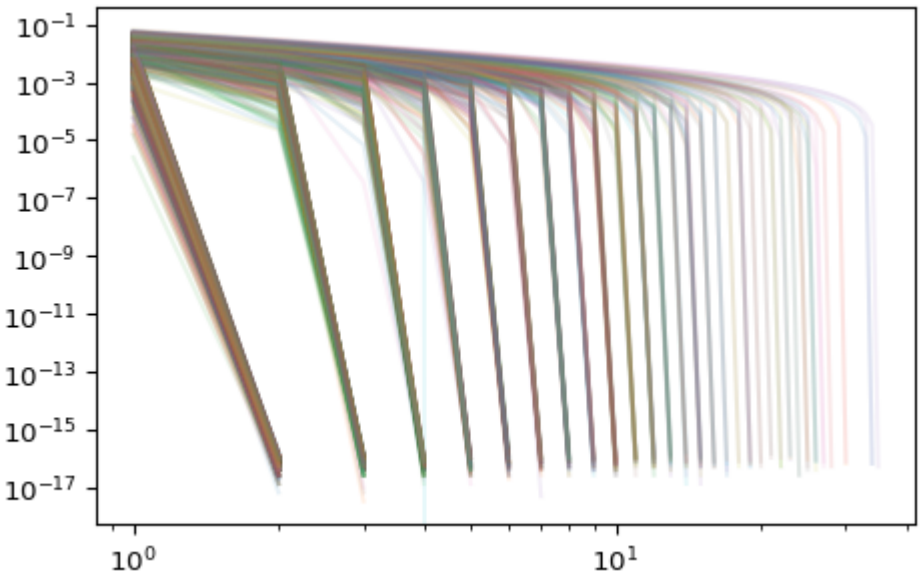

Figure 3: Interesting trends in reduction in error with increased sample size iteration.

The rigorous minimal packing may be significantly larger than the first-order approximation or some other packing set within some acceptable tolerance. Whether a rigorous or approximate solution is needed depends very much on the application. But these concepts can be used to efficiently quantify the computational cost of rigor in the $S : G$ match.

Insert a plot showing a realistic GSD and the sieve-specific error trend with increasing N. I could draw a typical GSD with a dual axis with N on the right. There would be several size-N lines with a colorscale indicating their error? or I could plot several GSDs and color by N and use the other axis for the error.

They can also be used to evaluate the computational cost of approximate and rigorous minimal packings for different USCS classifications.

Insert a plot showing error v. N for granular USCS soils.

Bibliography