https://doi.org/10.5194/egusphere-2023-1690
Preprint. Discussion started: 18 September 2023

NorSand4AI: A Comprehensive Triaxial Test Simulation Database for NorSand Constitutive Model Materials

Luan Carlos de Sena Monteiro Ozelim[1], Michéle Dal Toé Casagrande[1], and André Luís Brasil Cavalcante[1]

[1]Department of Civil and Environmental Engineering, University of Brasilia,Campus Universitário Darcy Ribeiro, SG12, Asa Norte. 70910-900, Brasilia-DF, Brazil

Correspondence: Luan Carlos de Sena Monteiro Ozelim (luanoz@gmail.com, ozelim@unb.br)

Abstract. To learn, humans observe and experience the world, collect data, and establish patterns through repetition. In scientific discovery, these patterns and relationships are expressed as laws and equations, data as properties and variables, and observations as events. Data-driven techniques aim to provide an impartial approach to learning using raw data from actual or simulated observations. In soil science, parametric models known as constitutive models are used to represent the behavior of natural and artificial materials. Creating data-driven constitutive models using deep learning techniques requires large and consistent datasets, which are challenging to acquire through experiments. Synthetic data can be generated using a theoretical function, but there is a lack of literature on high-volume and robust datasets of this kind. Digital soil models can be utilized to conduct numerical simulations that produce synthetic results of triaxial tests, which are regarded as the preferred tests for assessing soil's constitutive behavior. Due to its limitations for modeling real sands, the Modified Cam Clay model has been replaced by the NorSand model in some situations where sand-like materials need to be modelled. Therefore, for a material following the NorSand model, the present paper presents a first-of-its-kind database that addresses the size and complexity issues of creating synthetic datasets for nonlinear constitutive modeling of soils by simulating both drained and undrained triaxial tests of 2000 soil types, each subjected to 40 initial test configurations, resulting in a total of 160000 triaxial test results. Each simulation dataset comprises a 4000×10 matrix that can be used for general multivariate forecasting benchmarks, in addition to direct geotechnical and soil science applications.

1 Introduction

Jefferies and Been (2015) argue that the time and effort required to create models tailored to a specific project present limitations to the use of more comprehensive numerical analyses in engineering practice. This is because the cost of developing customized computational tools (such as implementation numerical solvers or refined constitutive models) can quickly exhaust the project's available budget. Therefore, it is important to find ways to create or modify models that can accommodate the unique characteristics of the materials of interest without necessitating elaborate computational implementations.

It is precisely in this context that the creation of data-driven and physically-informed metamodels emerges. These metamodels, when based on artificial intelligence techniques, especially machine learning (ML) and deep learning (DL), may be able to provide accurate and computationally cheap models, allowing them to be a perfect link between complex computational models and real-time data collection and monitoring.

Montáns et al. (2019) emphasize that human learning involves observing and experiencing the world, collecting data, and identifying patterns through repeated experiments. Scientific discovery involves formalizing these patterns and relationships into laws and equations, transforming data into properties and variables, and converting observations into events. Although laws and equations aid learning, the classical learning process in science is often slow and expensive, requiring extensive observation and experimentation to understand the main variables and their impact on the phenomenon.

Data-driven procedures, on the other hand, seek, if possible, an implicitly unbiased approach to our learning experience based on raw data from actual or synthetic observations. These procedures have the added advantage of testing correlations between different variables and observations, learning unanticipated patterns in nature, and allowing us to discover new scientific laws or even make predictions without the availability of such laws (Montáns et al., 2019).

The recent rapid increase in the availability of measurement data from physical systems as well as from massive numerical simulations has stimulated the development of many data-driven methods for modeling and predicting dynamics. At the forefront of data-driven methods are Deep Neural Netwroks (DNNs). DNNs not only achieve superior performance for tasks such as image classification, but have also proven effective for future state prediction of dynamical systems (Haghighat et al., 2021). A key limitation of DNNs, and similar data-based methods, is the lack of interpretability of the resulting model: they are focused on prediction and do not provide governing equations or clearly interpretable models in terms of the original set of variables. An alternative data-based approach uses symbolic regression to directly identify the structure of a nonlinear dynamical system from data (Schmidt and Lipson, 2009). This works remarkably well for discovering interpretable physical models, but symbolic regression is computationally expensive and can be difficult to scale to large problems (Montáns et al., 2019).

In order to create metamodels from Neural Networks (NN), this type of approach generally requires a priori calibration of the algorithms from data considered to be representative of material behavior (He et al., 2021). For example, NNs have been applied to model a variety of materials, including concrete materials (Ghaboussi et al., 1991), hyperelastic materials (Shen et al., 2005), viscoplastic steel material (Furukawa and Yagawa, 1998), and homogenized properties of mixed structures (Lefik and Schrefler, 2003). Once calibrated, NN-based constitutive models have been integrated into finite element codes to predict path- or rate-dependent material behaviors (Lefik and Schrefler, 2003; Hashash et al., 2004; Jung and Ghaboussi, 2006; Stoffel et al., 2019).

Recently, deep neural networks with special mechanistic architectures, such as Recurrent Neural Networks (RNNs), have been applied to path-dependent materials (Wang and Sun, 2018; Mozaffar et al., 2019; Heider et al., 2020). It is clear that this type of approach has found significant applications in a wide range of engineering fields, as reinforced by He et al. (2021), when they argue that data-driven computation with physical constraints is an emerging computational paradigm that allows the simulation of complex materials directly based on the materials database and disregards the classical constitutive model construction.

To develop a data-driven constitutive model, a substantial and reliable dataset is necessary. However, obtaining a sufficiently large dataset for soil science can be challenging since experimental data is often limited and inadequate for training ML and

DL algorithms. Generating synthetic data using a theoretical function can be a useful alternative, as it allows for the creation of an unlimited supply of data (Zhang et al., 2021a).

The literature suggests that data-driven models should initially be developed using synthetic datasets to establish a general framework, which can later be applied to experimental datasets to enhance the model's robustness and aid in discovering potential mechanisms of soil behavior (Zhang et al., 2021a). By calibrating constitutive models on synthetic datasets, the impact of experimental and measurement errors on the mapping ability of machine learning algorithms can be eliminated (Zhang et al., 2020). Therefore, creating large and reliable synthetic datasets is a crucial step in constructing data-driven constitutive models.

Currently, there is a lack of robust and high-volume datasets in the literature for soil modeling tasks. One effective method to generate synthetic datasets is through numerical simulations performed on digital soil models. Typically, these simulations involve selecting a parametric constitutive model, sampling some parameters, and running simulations that mimic real-world test setups. In soil modeling, triaxial tests are commonly simulated using conventional physics-driven constitutive models, such as simple monotonic Konder's expression (Basheer, 2000) or more advanced models like the Modified Cam Clay (MCC) (Fu et al., 2007; Zhang et al., 2023).

In particular, a simple sand shear constitutive model was used to generate synthetic datasets in the work of Zhang et al. (2021b). A total of fourteen curves were generated to develop the ML-based constitutive model (nine curves for training and five curves for testing).

On the other hand, the MCC constitutive model was utilized to produce a benchmark stress–strain dataset of a virtual soil in the work of Zhang et al. (2023). In that study, a total of 250 soil types were considered, with 125 being part of the training dataset and the remaining 125 in the testing dataset. Considering all the initial states in the paper by Zhang et al. (2023), 1125 sets of stress–strain samples were employed as the training dataset, while 1250 sets of stress–strain samples constituted the testing dataset.

The MCC model has been a fundamental element in numerous complex models developed in recent times (Yao et al., 2008). However, this model and its variations are not well-suited for depicting the behavior of actual sands due to their insufficient representation of key features such as yielding and dilation. This is because these models assume that soils denser than the critical state line are over-consolidated, resulting in an unrealistically high stiffness and excessively exaggerated strength (Woudstra, 2021).

In this scenario, the NorSand model (Jefferies, 1993) emerges as a suitable alternative due to its relatively simple critical state formulation and low number of input parameters. This model is a generalized critical state model based on the state parameter ψ, as defined by Jefferies (1993):

$$\psi = e - e_c \tag{1}$$

where e is the current void ration and e_c is the void ratio at the critical state (Jefferies, 1993). The NorSand model emulates natural soil behavior by incorporating associated plasticity and limited hardening, which enables dilation similar to that observed in real soils. This limited hardening causes yielding during unloading conditions and provides second-order detail in replicating observed soil behavior (Silva et al., 2022; Jefferies and Been, 2015).

Thus, the current paper aims to address two main issues: the quantity and complexity of synthetic datasets for nonlinear constitutive modeling of soils. The first aspect is addressed by simulating both drained and undrained triaxial tests of 2000 soil types, each is subjected to 40 initial test configurations. As a result, a total of 160000 triaxial test results for soils following the NorSand model are made available in this paper.

A comprehensive database like this is crucial for developing ML and artificial intelligence models of geotechnical materials. We are confident that this database will be widely used by both academic and industry communities. Furthermore, researchers interested in modeling sequential data, such as time series, could use this dataset for benchmarking purposes, as the highly non-linear nature of the constitutive model poses a significant challenge to ML and DL techniques.

2 Methods

The NorSandTXL program is an Excel spreadsheet with all coding in the VBA environment and can be downloaded at http://www.crcpress.com/product/isbn/9781482213683, as indicated in the book by Jefferies and Been (2015). This particular spreadsheet simulates drained and undrained triaxial tests of materials governed by the NorSand constitutive model.

The input features available in NorSandTXL are presented in Table 1

In order to massively simulate triaxial test conditions for materials following the NorSand constitutive model, a Python routine has been developed. This routine performs two main steps: sampling and simulation.

For the sampling process, all 14 input parameters are sampled using the Latin hypercube sampling algorithm. This sampling process must to be nested, as there are two levels of hierarchy in the parameters: the higher level deals with the soil properties, which are unique for a given material, while the lower level considers the initial soil state during the triaxial tests. As a result, the sampling process needs to: a) account for different types of materials and b) for each type of material, consider several testing conditions.

Thus, the following sampling procedure is considered to account for n_{soils} types of soils under $n_{conditions}$ initial testing conditions:

– Sample the soil properties (the first ten parameters in Table 1), obtaining a vector of properties sp_i, $i = 1,...,n_{soils}$, such that $sp_i \in \mathbb{R}^{10}$. The sampling is performed using the centered Latin hypercube sampling algorithm implemented in the *chaospy* package (Feinberg and Langtangen, 2015) with a maximin criterion.

– For each sp_i, the initial testing conditions (the last four parameters in Table 1) are sampled using the standard Latin hypercube sampling algorithm implemented in the *chaospy* package (Feinberg and Langtangen, 2015) with a ratio criterion. This way, the vectors $ic_{i,j} \in \mathbb{R}^4$, $j = 1,...,n_{conditions}$ are obtained for each sp_i. The maximum value of ψ_0 is set to $\psi_{max}/5$ (as indicated in Table 1) for numerical stability. Additionally, to make the $ic_{i,j}$ different for each sp_i, the random seed of the sampling algorithm is changed for each i.

Soil properties			
Parameter Class	Parameter	Sampling range	Units
CSL parameters	$\Gamma\|_{p=1kPa}$	[0.9,1.4]	-
	λ	[0.01,0.07]	$(\ln \mathrm{kPa})^{-1}$
Plasticity	M_{tc}	[1.2,1.5]	-
	N	[0.2,0.5]	-
	χ_{tc}	[2,5]	-
	H_0	[75,500]	-
	H_ψ	[200,500]	-
Elasticity	$G_{max}\|_{p0}$	[30,100]	*MPa*
	G_{exp}	[0.1,0.6]	-
	ν	[0.1,0.3]	-
Initial Soil State			
Parameter Class	Parameter	Sampling range	Units
Stress and Deformability	ψ_0	[-0.2,$\psi_{max}/5$]	-, where $\psi_{max} = M_{tc}/(\chi(1+N))$
	p_0	[50,1000]	kPa
	K_0	[0.8,1.2]	-
	OCR ("R")	[0.5,3]	-

Table 1. Input values for NorSandTXL

From the procedure above, the matrix In of input parameters is obtained, whose rows are NorSandTXL input vectors obtained by concatenating each sp_i with all the $ic_{i,j}$, i.e., $[concat(sp_1, ic_{1,1}), concat(sp_1, ic_{1,2}), ..., concat(sp_{n_{soils}}, ic_{n_{soils},n_{conditions}})]$, where $concat$ denotes a concatenation operation between vectors. This implies that In is a $(n_{soils}n_{conditions})$ by 14 matrix.

The simulation step, on the other hand, involves opening the Excel spreadsheet provided in the book by Jefferies and Been (2015), inputting the sampled parameters, running both drained and undrained simulations for the input parameters and collecting their respective results, saving them in *.h5* format files for posterior processing.

The file extension *.h5* is associated with the Hierarchical Data Format (HDF5) (The HDF Group, 1997-2023), which is a type of high-performance distributed file system. It is specifically designed to manage large and complex data sets efficiently and flexibly. Additionally, it enables a self-describing file format that is portable and supports parallel I/O for data compression (Lee et al., 2022), and has shown superior performance with high-dimensional and highly structured data (Nti-Addae et al., 2019). Literature indicates that the HDF5 has been popular in scientific communities since the late 1990s (Lee et al., 2022), which is evident by the large number of open-source and commercial software packages for data visualization and analysis that can read and write HDF5 (Group, Accessed on April 24, 2023). As a result, this is the data format chosen for the present paper.

3 Data Records

In the present paper, $n_{soils} = 2000$ and $n_{conditions} = 40$. Additionally, the folder containing the produced dataset can be found in Ozelim et al. (2023) and has the following structure:

NorSandTXL_H5 \Simus**TT**\Par_**X**_**Y**.h5

where **TT** stands for the test type (Drained or Undrained), **X** is the material index (from 0 to 1999) and **Y** is the sequential index for the input parameters (from 0 to 79999).

Each *Par_X_Y.h5* file contains a dataset titled 'NorSandTXL' which includes the simulation results. By design, the NorSandTXL Excel spreadsheet considers 4000 strain steps to go from zero to approximately 20% nominal axial strain at the end of the simulated test. The authors of the spreadsheet indicate that this amount is both convenient and sufficient (Jefferies and Been, 2015). On the other hand, for a triaxial effective stress state with vertical stress σ'_a (kPa) and confining stress σ'_r (kPa), a total of 10 entities are reported from the tests, which are: ϵ_1 (axial strain); ϵ_v (volumetric strain); $p' = (\sigma'_a + 2\sigma'_r)/3$ (mean effective stress in kPa); $q = \sigma'_a - \sigma'_r$ (deviatoric stress in kPa); e (void ration); p_i/p (stress ratio); $(p_i/p)_{max}$ (maximum stress ratio); ψ (state parameter); D_p (dilation) and $\eta = q/p'$. Thus, the dataset is a 4000×10 array, as presented in Table 2.

ϵ_1	ϵ_v	p	q	e	p_i/p	$(p_i/p)_{max}$	ψ	D_p	η
0	0	200	0	0.9021	0.42306	1	0	0.92603	0
0.06097	0.04314	209.561	28.2795	0.90128	0.40376	1	0	0.92603	0.13495
0.07544	0.05481	210.703	31.7059	0.90106	0.40811	0.99319	0.001981083	1.31505	0.15048
0.0897	0.06628	211.821	35.0611	0.90084	0.41236	0.99284	0.002085266	1.29952	0.16552
					...				
19.3293	2.10004	387.564	562.29	0.86216	1.00101	1.00087	-0.000251146	-0.00146	1.45083
19.3334	2.10003	387.564	562.29	0.86216	1.00101	1.00087	-0.000251018	-0.00146	1.45083
19.3374	2.10002	387.564	562.29	0.86216	1.00101	1.00087	-0.000250889	-0.00146	1.45083

Table 2. The 'NorSandTXL' dataset present in each *Par_X_Y.h5* file.

It is worth noticing that the values stored are of the type *float32*, which is sufficient for the applications envisioned for the dataset. In addition to the simulation results, the dataset also contains the attributes shown in Table 3. The correspondence between the attributes, whose data type is either *float32* or *<U7* (fixed-length character string of 7 Unicode characters), and NorSandTXL input parameters is also presented in Table 3.

It is easy to see that the dataset attributes in each file allow for a complete reproduction of the results, if desired. The units of the parameters are consistent with NorSandTXL, as presented in Table 1.

Attribute	Parameter/Value
'Gamma'	$\Gamma\|_{p=1kPa}$
'lambda'	λ
'Mtc'	M_{tc}
'N'	N
'Xtc'	χ_{tc}
'H0'	H_0
'Hy'	H_ψ
'Gmax_p0'	$G_{max}\|_{p_0}$
'G_exp'	G_{exp}
'n'	ν
'Psi_0'	ψ_0
'p0'	p_0
'K0'	K_0
'OCR'	OCR ("R")
'Type'	Drained or Undrained

Table 3. Attributes of the 'NorSandTXL' dataset present in each *Par_**X**_**Y**.h5* file.

4 Technical Validation

Considering that the engine running the triaxial test simulations is the Excel spreadsheet presented in the book by Jefferies and Been (2015) and that such spreadsheet has been extensively validated by both academia and industry, there is no need to discuss the technical quality of the dataset.

5 Usage Notes

In Python, the *h5py* package provides all the necessary tools to interact with the *.h5* files produced and made available in the NorSand4AI dataset. Depending on the intended application, it might be beneficial to down-sample the 4000×10 matrix to increase the axial strain increments. This can be accomplished using standard Python packages such as *pandas* and *numpy*.

6 Conclusions

Obtaining massive datasets for modelling the behavior of soils is of great interest, not only because new artificial intelligence algorithms can be built, but also to assess the adequacy of newly proposed physically informed models. In the context of critical state approaches, the NorSand model has shown provide a good balance balance complexity and accuracy. Also, this model is used to assess the liquefaction potential of soils, which is a major cause of high scale disasters lately, such as tailing dams' failures.

This way, in the present paper a massive dataset with 160000 triaxial tests has been obtained by combining an Excel spreadsheet (and its Visual Basic programs) with Python. This first-of-its-kind database addresses the size and complexity issues of creating synthetic datasets for nonlinear constitutive modeling of soils by simulating both drained and undrained triaxial tests of 2000 soil types, each subjected to 40 initial test configurations. Each simulation dataset comprises a 4000×10 matrix that can be used for general multivariate forecasting benchmarks, in addition to direct geotechnical and soil science applications. Both the dataset and the codes elaborated to generate it have been provided.

7 Code and data availability

All data associated with the current submission is available at Ozelim et al. (2023). Any updates will also be published on Zenodo, and the final DOI cited in the manuscript. The Python code used to generate the NorSandAI dataset is described in the Appendix of the present paper.

Appendix A: Python code description

At first, the following Python packages need to be imported:

```
import numpy as np
import pandas as pd
import xlwings as xw
import string
from skopt.space import Space
from skopt.sampler import Lhs
import os
import math
import h5py
```

The packages *numpy*, *math* and *pandas* are required for data manipulation and numeric calculations. The *xlwings* package is needed to bridge Python and Excel. On the other hand, the *string* package is necessary to convert the (row-column) positional encoding to the (row-letter) alphanumeric encoding used in Excel. For the sampling procedure, *skopt* is required. Lastly, for creating folders and files, both *os* and *h5py* should be imported.

Let $dictpos$ be a dictionary that points to the locations in the spreadsheet of the cells corresponding to each input parameter. Additionally, let $dict\_ranges\_material$ and $dict\_ranges\_test$ be dictionaries specifying the sampling ranges of the input parameters. For this paper, these dictionaries are:

```
dictpos = {"Gamma":[6,4],"lambda":[7,4],"Mtc":[14,4], "N":[15,4],
2:          "Xtc": [16,4],"H0":[17,4],"Hy":[18,4], "Gmax_p0":[21,4],
3:          "G_exp": [22,4], "nu":[23,4],"Psi_0":[27,4],"p0":[29,4],
4:          "K0": [30,4], "OCR": [32,4]}
5: dict_ranges_material = {"Gamma":[0.9,1.4],"lambda":[0.01,0.07],"Mtc":[1.2,1.5],"N":[0.2,0.5],
6:          "Xtc": [2,5],"H0":[75,500],"Hy":[200,500], "Gmax_p0":[30,100],
7:          "G_exp": [0.1,0.6], "nu":[0.1,0.3]}
8: dict_ranges_test = {"Psi_0":[−0.2,0.2],"p0":[50,1000],
9:          "K0": [0.8,1.2], "OCR": [0.5,3]}
```

To generate the inputs for the NorSandTXL spreadsheet, considering $n\_samples$ soil types and $n\_samples\_2$ initial test conditions, the following code was considered:

```
1: def gen_NorSand_par_2(dict_ranges_material,dict_ranges_test,n_samples,n_samples_2):
2:   lhs = Lhs(lhs_type="centered", criterion='maximin')
3:   lhsinner = Lhs(criterion="ratio")
4:   space_material = Space([(0, 1.) for x in range(len(dict_ranges_material))])
5:   space_test = Space([(0, 1.) for x in range(len(dict_ranges_test))])
6:   x_mat = lhs.generate(space_material.dimensions, n_samples,random_state=11)
7:   data_inp_mat = (np.array(x_mat).T)
8:   data_expand_mat = []
9:   for ind_vals in range(len(dict_ranges_material)):
10:     vlow,vup = list(dict_ranges_material.values())[ind_vals]
11:     data_pts = data_inp_mat[ind_vals]
12:     data_expand_mat.append((vup−vlow)∗data_pts + vlow)
13:   data_expand_mat = np.round(np.array(data_expand_mat),4)
14:   data_expand_tst_corretos=[]
15:   for pbb,yv in enumerate(data_expand_mat.T):
16:     x_tst = lhsinner.generate(space_test.dimensions, n_samples_2,random_state=int(11+2∗pbb))
17:     data_inp_tst = (np.array(x_tst).T)
18:     data_expand_tst = []
19:     for ind_vals in range(len(dict_ranges_test)):
20:       if ind_vals==0:
21:         data_expand_tst.append(data_inp_tst[ind_vals])
22:       else:
23:         vlow,vup = list(dict_ranges_test.values())[ind_vals]
24:         data_pts = data_inp_tst[ind_vals]
25:         data_expand_tst.append((vup−vlow)∗data_pts + vlow)
26:     data_expand_tst = np.array(data_expand_tst)
27:     data_expand_tst_prov = data_expand_tst.copy()
```

```
        data_expand_tst_prov[0] = np.array([(np.clip(yv[2]/(yv[4]∗(1+yv[3])),0, yv[2]/(5∗yv[4]∗(1+yv[3])))+0.2)∗lhsv−0.2 for lhsv in
         data_expand_tst_prov[0]])
        data_expand_tst_corretos.append(data_expand_tst_prov)
    data_expand_tst_corretos = np.round(np.array(data_expand_tst_corretos),4)
    final_comp=[]
    for mat_vals,tst_vals in zip(data_expand_mat.T,data_expand_tst_corretos):
        for ti_vals in tst_vals.T:
            final_comp.append(np.concatenate((mat_vals,ti_vals),axis=0))
    return final_comp
```

On the other hand, to run the NorSandTXL Excel spreadsheet located in $path\_xlsm$ for all the input parameters previously obtained as $final\_comp = gen\_NorSand\_par\_2(\, dict\_ranges\_material, dict\_ranges\_test, n\_samples, n\_samples\_2)$, the following function can be run.

```
def run_NorSand_simus_P(final_comp,dictpos,n_samples_2,path_xlsm):
    letras = list(string.ascii_uppercase)
    wb = xw.Book(path_xlsm)
    app = wb.app
    macro_vba = app.macro("'NorTxl.xlsm'!RunSim")
    macro_vba_type = app.macro("'NorTxl.xlsm'!ChangeSimMode")
    ws = wb.sheets["Params␣&␣Plots"]
    results_comp = []
    for idini,new_v in enumerate(final_comp):
        matv = int(math.floor(idini/n_samples_2))
        for nv,ps in zip(new_v,dictpos.values()):
            pl,pc = ps
            pfinal = letras[pc−1]+str(pl)
            ws[pfinal].value = nv
        for type_v in ["Drained","Undrained"]:
            if ws["D34"].value == type_v:
                pass
            else:
                macro_vba_type()
            macro_vba()
            ws_results = wb.sheets["Txl␣SimResults"]
            np_arr = (ws_results['A4'].expand('table')).value
            path_xlsm_init = ("\\").join(path_xlsm.split("\\")[:−1])
            new_h5_file = path_xlsm_init+'\\Simus\\'+type_v+"\\Par_"+str(matv)+"_"+str(idini)+".h5"
            new_h5_file_spl = new_h5_file.split("\\")
            for va in range(−3,0):
                try:
                    os.mkdir(os.path.join(*new_h5_file_spl[:va]))
                except:
                    pass
            h5f = h5py.File(new_h5_file, 'w')
            dd = h5f.create_dataset('NorSandTXL', data=np.array((ws_results['A4'].expand('table')).value).astype(np.float32),compression
        ='gzip')
            for keyv,pvalu in zip(dictpos.keys(),new_v.astype(np.float32)):
                dd.attrs[keyv] = pvalu
            dd.attrs["Type"] = type_v
            h5f.close()
```

The function $run\_NorSand\_simus\_P$ runs the simulation and also saves the results as *.h5* files in the same folder as the Excel spreadsheet. In this case, the new files are saved following the naming convention and folder structure discussed in the paper.

Author contributions. Conceptualization, L.C.d.S.M.O.; methodology, L.C.d.S.M.O.; software, L.C.d.S.M.O.; validation, L.C.d.S.M.O.; formal analysis, L.C.d.S.M.O., M.D.T.C. and A.L.B.C.; investigation, L.C.d.S.M.O.; writing—original draft preparation, L.C.d.S.M.O.; writing—review and editing, M.D.T.C. and A.L.B.C.; supervision, M.D.T.C.; funding acquisition, L.C.d.S.M.O. All authors have read and agreed to the published version of the manuscript.

Competing interests. The authors declare no competing interests.

Acknowledgements. The authors acknowledge the support of the National Council for Scientific and Technological Development (CNPq Grant 102414/2022-0). This study was also financed in part by the Coordination for the Improvement of Higher Education Personnel (CAPES)—Finance Code 001.

References

Basheer, I. A.: Selection of Methodology for Neural Network Modeling of Constitutive Hystereses Behavior of Soils, Computer-Aided Civil and Infrastructure Engineering, 15, 445–463, https://doi.org/10.1111/0885-9507.00206, 2000.

Feinberg, J. and Langtangen, H. P.: Chaospy: An open source tool for designing methods of uncertainty quantification, Journal of Computational Science, 11, 46–57, https://doi.org/https://doi.org/10.1016/j.jocs.2015.08.008, 2015.

Fu, Q., Hashash, Y. M., Jung, S., and Ghaboussi, J.: Integration of laboratory testing and constitutive modeling of soils, Computers and Geotechnics, 34, 330–345, https://doi.org/10.1016/j.compgeo.2007.05.008, 2007.

Furukawa, T. and Yagawa, G.: Implicit constitutive modelling for viscoplasticity using neural networks, International Journal for Numerical Methods in Engineering, 43, 195–219, https://doi.org/10.1002/(sici)1097-0207(19980930)43:2<195::aid-nme418>3.0.co;2-6, 1998.

Ghaboussi, J., Garrett, J. H., and Wu, X.: Knowledge-Based Modeling of Material Behavior with Neural Networks, Journal of Engineering Mechanics, 117, 132–153, https://doi.org/10.1061/(asce)0733-9399(1991)117:1(132), 1991.

Group, T. H.: Software Using HDF5, https://portal.hdfgroup.org/display/support/Software+Using+HDF5, Accessed on April 24, 2023.

Haghighat, E., Raissi, M., Moure, A., Gomez, H., and Juanes, R.: A physics-informed deep learning framework for inversion and surrogate modeling in solid mechanics, Computer Methods in Applied Mechanics and Engineering, 379, 113 741, https://doi.org/10.1016/j.cma.2021.113741, 2021.

Hashash, Y. M. A., Jung, S., and Ghaboussi, J.: Numerical implementation of a neural network based material model in finite element analysis, International Journal for Numerical Methods in Engineering, 59, 989–1005, https://doi.org/10.1002/nme.905, 2004.

He, X., He, Q., and Chen, J.-S.: Deep autoencoders for physics-constrained data-driven nonlinear materials modeling, Computer Methods in Applied Mechanics and Engineering, 385, 114 034, https://doi.org/10.1016/j.cma.2021.114034, 2021.

Heider, Y., Wang, K., and Sun, W.: SO(3)-invariance of informed-graph-based deep neural network for anisotropic elastoplastic materials, Computer Methods in Applied Mechanics and Engineering, 363, 112 875, https://doi.org/10.1016/j.cma.2020.112875, 2020.

Jefferies, M. and Been, K.: Soil Liquefaction: A Critical State Approach, Second Edition, CRC Press, 2 edn., https://doi.org/10.1201/b19114, 2015.

Jefferies, M. G.: Nor-Sand: a simle critical state model for sand, Géotechnique, 43, 91–103, https://doi.org/10.1680/geot.1993.43.1.91, 1993.

Jung, S. and Ghaboussi, J.: Neural network constitutive model for rate-dependent materials, Computers & Structures, 84, 955–963, https://doi.org/10.1016/j.compstruc.2006.02.015, 2006.

Lee, S., yuan Hou, K., Wang, K., Sehrish, S., Paterno, M., Kowalkowski, J., Koziol, Q., Ross, R. B., Agrawal, A., Choudhary, A., and keng Liao, W.: A case study on parallel HDF5 dataset concatenation for high energy physics data analysis, Parallel Computing, 110, 102 877, https://doi.org/https://doi.org/10.1016/j.parco.2021.102877, 2022.

Lefik, M. and Schrefler, B.: Artificial neural network as an incremental non-linear constitutive model for a finite element code, Computer Methods in Applied Mechanics and Engineering, 192, 3265–3283, https://doi.org/10.1016/s0045-7825(03)00350-5, 2003.

Montáns, F. J., Chinesta, F., Gómez-Bombarelli, R., and Kutz, J. N.: Data-driven modeling and learning in science and engineering, Comptes Rendus Mécanique, 347, 845–855, https://doi.org/10.1016/j.crme.2019.11.009, 2019.

Mozaffar, M., Bostanabad, R., Chen, W., Ehmann, K., Cao, J., and Bessa, M. A.: Deep learning predicts path-dependent plasticity, Proceedings of the National Academy of Sciences, 116, 26 414–26 420, https://doi.org/10.1073/pnas.1911815116, 2019.

Nti-Addae, Y., Matthews, D., Ulat, V. J., Syed, R., Sempéré, G., Pétel, A., Renner, J., Larmande, P., Guignon, V., Jones, E., and Robbins, K.: Benchmarking database systems for Genomic Selection implementation, Database, 2019, baz096, https://doi.org/10.1093/database/baz096, 2019.

Ozelim, L. C. d. S. M., Casagrande, M. D. T., and Cavalcante, A. L. B.: NorSand4AI: A Comprehensive Triaxial Test Simulation Database for NorSand Constitutive Model Materials, https://doi.org/10.5281/zenodo.8170537, 2023.

Schmidt, M. and Lipson, H.: Distilling Free-Form Natural Laws from Experimental Data, Science, 324, 81–85, https://doi.org/10.1126/science.1165893, 2009.

Shen, Y., Chandrashekhara, K., Breig, W., and Oliver, L.: Finite element analysis of V-ribbed belts using neural network based hyperelastic material model, International Journal of Non-Linear Mechanics, 40, 875–890, https://doi.org/10.1016/j.ijnonlinmec.2004.10.005, 2005.

Silva, J. P., Cacciari, P., Torres, V., Ribeiro, L. F., and Assis, A.: Behavioural analysis of iron ore tailings through critical state soil mechanics, Soils and Rocks, 45, 1–13, https://doi.org/10.28927/sr.2022.071921, 2022.

Stoffel, M., Bamer, F., and Markert, B.: Neural network based constitutive modeling of nonlinear viscoplastic structural response, Mechanics Research Communications, 95, 85–88, https://doi.org/10.1016/j.mechrescom.2019.01.004, 2019.

The HDF Group: Hierarchical Data Format, version 5, https://www.hdfgroup.org/HDF5/, 1997-2023.

Wang, K. and Sun, W.: A multiscale multi-permeability poroplasticity model linked by recursive homogenizations and deep learning, Computer Methods in Applied Mechanics and Engineering, 334, 337–380, https://doi.org/10.1016/j.cma.2018.01.036, 2018.

Woudstra, L.-J.: Verification, Validation and Application of the NorSand Constitutive Model in PLAXIS: Single-stress point analyses of experimental lab test data and finite element analyses of a submerged landslide, Master's thesis, TU Delft Civil Engineering & Geosciences, 2021.

Yao, Y., Sun, D., and Matsuoka, H.: A unified constitutive model for both clay and sand with hardening parameter independent on stress path, Computers and Geotechnics, 35, 210–222, https://doi.org/10.1016/j.compgeo.2007.04.003, 2008.

Zhang, N., Zhou, A., Jin, Y.-F., Yin, Z.-Y., and Shen, S.-L.: An enhanced deep learning method for accurate and robust modelling of soil stress–strain response, Acta Geotechnica, https://doi.org/10.1007/s11440-023-01813-8, 2023.

Zhang, P., Yin, Z.-Y., Jin, Y.-F., and Ye, G.-L.: An AI-based model for describing cyclic characteristics of granular materials, International Journal for Numerical and Analytical Methods in Geomechanics, 44, 1315–1335, https://doi.org/10.1002/nag.3063, 2020.

Zhang, P., Yin, Z.-Y., and Jin, Y.-F.: State-of-the-Art Review of Machine Learning Applications in Constitutive Modeling of Soils, Archives of Computational Methods in Engineering, 28, 3661–3686, https://doi.org/10.1007/s11831-020-09524-z, 2021a.

Zhang, P., Yin, Z.-Y., Jin, Y.-F., and Liu, X.-F.: Modelling the mechanical behaviour of soils using machine learning algorithms with explicit formulations, Acta Geotechnica, 17, 1403–1422, https://doi.org/10.1007/s11440-021-01170-4, 2021b.