MethodsX 6 (2019) 2118–2126

Contents lists available at ScienceDirect

MethodsX

journal homepage: www.elsevier.com/locate/mex

Method Article

SPorDyn: A Python code for modeling the evolution of soil pore size distribution after tillage

Parvathy Chandrasekhar[a,b,*], Janis Kreiselmeier[a,b], Andreas Schwen[c], Thomas Weninger[c], Stefan Julich[b], Karl-Heinz Feger[b], Kai Schwärzel[a,d]

[a] United Nations University Institute for Integrated Management of Material Fluxes and of Resources (UNU-FLORES), Ammonstraße 74, 01067 Dresden, Germany
[b] Institute of Soil Science and Site Ecology, Technische Universität Dresden, Pienner Straße 19, 01737 Tharandt, Germany
[c] Institute for Soil Physics and Rural Water Management, University of Natural Resources and Life Sciences, Muthgasse 18, 1190 Vienna, Austria
[d] Institute of Forest Ecosystems, Thünen Institute, Alfred-Möller Straße 1, 16225 Eberswalde Germany

A B S T R A C T

Surface soil structure is very responsive to natural and anthropogenic impacts and these changes alter soil hydraulic properties and the soil water budget. In the midst of a dearth of efforts to capture soil structural dynamics, an analytical solution to the Fokker-Planck Equation with physically-based coefficients has shown promising results in predicting the evolution of soil pore space in agricultural soils. In this study, the Python code for the analytical solution is shown along with steps to estimate coefficients leading towards obtaining the analytical solution.

- Python code for modeling the evolution of soil pore space based on an existing model is shared.
- The code for the estimation of physically-based coefficients of the model and parameter optimization are also shown.
- The final output of the model may be used in estimation of soil water retention and hydraulic conductivity functions.

* Corresponding author at: United Nations University Institute for Integrated Management of Material Fluxes and of Resources (UNU-FLORES), Ammonstraße 74, 01067 Dresden, Germany.
E-mail addresses: chandrasekhar@unu.edu (P. Chandrasekhar), kreiselmeier@unu.edu (J. Kreiselmeier), andreas.schwen@ages.at (A. Schwen), thomas.weninger@boku.ac.at (T. Weninger), stefan.julich@tu-dresden.de (S. Julich), karl-heinz.feger@tu-dresden.de (K.-H. Feger), schwaerzel@unu.edu (K. Schwärzel).

https://doi.org/10.1016/j.mex.2019.09.014

ARTICLE INFO
Method name: SPorDyn: Soil pore dynamics using Python
Keywords: Pore evolution model, Soil structure, Temporal dynamics, Tillage
Article history: Received 9 August 2019; Accepted 11 September 2019; Available online 13 September 2019

Specification Table

Subject Area:	• *Agricultural and Biological Sciences* • *Environmental Science*
More specific subject area:	*Modeling approach to capture the temporal dynamics of soil pore space*
Method name:	*SPorDyn: Soil pore dynamics using Python*
Name and reference of original method:	*Or, D., Leij, F.J., Snyder, V., Ghezzehei, T.A., 2000. Stochastic model for posttillage soil pore space evolution. Water Resour. Res. 36, 1641–1652.* *Leij, F.J., Ghezzehei, T.A., Or, D., 2002a. Modeling the dynamics of the soil pore-size distribution. Soil Tillage Res. 64, 61–78.* *Leij, F.J., Ghezzehei, T.A., Or, D., 2002b. Analytical models for soil pore-size distribution after tillage. Soil Sci. Soc. Am. J. 66, 1104–1114.* *Chandrasekhar,P., Kreiselmeier, J., Schwen,A., Weninger,T., Julich,S., Feger, K.-H., Schwärzel, K., 2019. K. Modeling the evolution of soil structural pore space in agricultural soils following tillage. Geoderma. 353. 401 - 414*
Resource availability:	*Code provided as supplementary material and as images in the text*

Method details

Surface soil structure is responsive to both natural and anthropogenic changes and consequently impacts soil hydraulic properties (SHP). Generally, the structural pore space with pore radius (r) $\geq$ 5 μm is expected to be the most affected by management practices such as tillage and natural stresses (e.g. rainfall). The need for inclusion of soil pore dynamics in hydrological models to improve their reliability and accuracy has been stressed in recent times [1–3]. In this context, Or et al. [4] proposed to use a partial differential equation (PDE) known as the Fokker-Planck Equation to capture the dynamics of soil pore size distribution (PSD) following tillage with respect to time and pore radius. Knowledge of the PSD will pave way to predict unsaturated SHP which can then be incorporated in hydrological models. The PDE comprises physically-based coefficients (drift (V), degradation (M) and dispersion (D) coefficients) and they encompass our perception of the mathematical behavior of soil PSD in response to tillage practices. The coefficients are subject to an initial condition as well as upper and lower boundary conditions. Based on these conditions, an analytical solution to the proposed PDE was provided by Leij et al. [5,6]. In Chandrasekhar et al. [1], the model was applied to different water retention parameter data sets around the world to evaluate its capability in predicting the temporal dynamics of soil pore space for two cases (1) when there is a change in the tillage regime/land-use and (2) in the months following tillage. In the present contribution, we share the Python code used in Chandrasekhar et al. [1] to capture the evolution of soil pore space following tillage.

This paper is organized as follows: As a first step, a skeletal framework of the mathematical model is briefly described. For a detailed overview of the model and its application, the reader is directed to Chandrasekhar et al. [1]. The required input data is listed. Brief descriptions of the remaining aspects of the model such as the coefficients as well as the accompanying code are then provided. Finally, the optimization process and steps to obtain the analytical solution are given.

Mathematical model

In this section, a skeletal overview of the mathematical model and its coefficients is provided. The following partial differential equation (PDE), also known as the Fokker-Planck equation, is used to describe the evolution of soil PSD following tillage [4]:

$$\frac{\partial f}{\partial t} = \frac{\partial}{\partial r}\left(D(r,t)\frac{\partial f}{\partial r}\right) - \frac{\partial}{\partial r}(V(r,t)f) - M(t)f \tag{1}$$

where f is the PSD or frequency [L^{-1}] of pores as a function of time t [T] and pore radius r [L], D the dispersion coefficient [$L^2\,T^{-1}$], V the drift coefficient [LT^{-1}] and M the degradation coefficient [T^{-1}]. D and V quantify the changes with time of the variance of the PSD and mean pore radius, respectively, while M is a first-order degradation factor representing instantaneous pore loss, i.e., the fraction of pores that are lost due to instantaneous collapse. The initial and boundary conditions for solving Eq. (1) are:

$$f(r,0) = f_0(r), \quad 0 < r < \infty \tag{2}$$

$$Vf - D\frac{\partial f}{\partial r} = 0, \quad r = 0, \quad t > 0 \tag{3}$$

$$\frac{\partial f}{\partial r} = 0, \quad r \to \infty, \quad t > 0 \tag{4}$$

f_0 in Eq. (2) is the initial PSD for the parameterization of which the lognormal distribution function of [7] is used. The lower boundary condition (Eq. 3) stipulates a zero-probability flux meaning that only positive pore sizes are allowed while the upper boundary condition (Eq. 4) necessitates a zero gradient for infinitely large pore radii. The PDE subject to the initial and boundary conditions yields the following analytical solution [5,6]:

$$\begin{aligned} f(r,T) = exp\left(\int_0^T \frac{M(\tau)}{V(\tau)}d\tau\right)\int_0^\infty f_0(\xi) \\ \times \left\{\frac{1}{\sqrt{4\pi\lambda T}}\left[exp\left(-\frac{(r-\xi+T)^2}{4\lambda T}\right) + exp\left(-\frac{r}{\lambda} - \frac{(r+\xi-T)^2}{4\lambda T}\right)\right]\right. \\ \left. + \frac{1}{2\lambda}exp\left(-\frac{r}{\lambda}\right)erfc\left(\frac{r+\xi-T}{\sqrt{4\lambda T}}\right)\right\}\, d\xi \end{aligned} \tag{5}$$

where τ and ξ are dummy integration variables. Due to the fact that tillage treatment cannot be readily converted to time as an independent variable, the cumulative drift term T is used as an independent variable instead of time, meaning that the evolution of PSD is predicted based on the gradual changes in the pore radii [5].

Input data

The input data for the modeling approach are water retention parameters obtained at different temporal stages. These parameters may be parameterized according to the Kosugi [7] water retention model. If Van Genuchten parameters [8] are available, they can be converted to the Kosugi parameters which is explained in Chandrasekhar et al. [1]. The required input water retention parameters are listed in Table 1.

Table 1
Input data: water retention parameters.

Parameter	Unit	Description
θ_s	[L^3 L^{-3}]	Saturated water content
θ_r	[L^3 L^{-3}]	Residual water content
σ	[-]	Standard deviation of the log-transformed pore radius
r_m	[L]	Median pore radius

```
#%% Import modules
import numpy as np
#=============================================================================================
# Function for initial pore size distribution using Kosugi's lognormal distribution function
def initial_PSD(r, TS, TR, sig, rm):
    """
    Parameters
    ----------
    r  : range of pore radii
    TS : saturated water content
    TR : residual water content
    sig: sigma
    rm : median pore radius

    Returns
    -------
    Pore size distribution at r
    """
    return (TS - TR)/(sig * np.sqrt(2*np.pi) * r) * (np.exp((-(np.log(r/rm))**2)/(2*sig**2)))
#=============================================================================================
```

Fig. 1. Code for initial pore size distribution.

Initial pore size distribution

Eq. (6) shows the lognormal distribution function for the pore radii [7]:

$$f_0(r) = \frac{\phi_0}{r\sigma\sqrt{2\pi}} \exp\left(-\frac{\left[\ln\left(\frac{r}{r_m}\right)\right]^2}{2\sigma^2}\right) \tag{6}$$

$$where \int_0^\infty f_0(r)dr = \phi_0, \quad 0 < r < \infty$$

where ϕ_0 [-] is the total initial porosity, r_m [L] is the initial median pore radius or geometric mean and σ [-] is the standard deviation of the log-transformed pore radius. Leij et al. [5] assumed ϕ_0 to be the difference between saturated and residual water contents: $\phi_0 = \theta_s - \theta_r$. Finally, r_m is calculated from the Young-Laplace equation where r = A/h, where A is a proportionality constant obtained from the variables of the equation, A = −0.149 cm^2 and h [L] is the pressure head. The code for the initial PSD is shown in Fig. 1.

Coefficients of the PDE

The coefficients of the PDE are calculated following the approach of [4] who used moment analysis of the PSD to yield the definitions for mean and variance. Moments are defined by integrating the PSD

```
#=============================================================================================
#%% Import modules
import numpy as np
from scipy import integrate

# Zero order moment
def PSD(r, TS, TR, sig, rm):
    """
    Parameters
    ----------
    r: range of pore radii
    TS: saturated water content
    TR: residual water content
    sig: sigma
    rm: median pore radius

    Returns
    -------
    Pore size distribution at r (after which the zero order moment can be obtained)
    """
    return (TS - TR)/(sig * np.sqrt(2*np.pi) * r) * (np.exp((-(np.log(r/rm))**2)/(2*sig**2)))

zero_moment = integrate.quad(zero_moment, 0, np.inf)[0]
#=============================================================================================
```

Fig. 2. Code for zero-order moment.

with respect to the pore size:

$$m_n(T) = \int_0^\infty r^n \, f(r,T)dr, \;\; n = 0,1,2 \tag{7}$$

Normalized moments (M_n) are calculated through division by the zero-order moment (m_0). The code to obtain the zero order moment is shown in Fig. 2.

Degradation term

The degradation term is expressed by means of an exponentially decaying function [9]:

$$M(t) = d \times \exp(ct), \quad c < 0 \tag{8}$$

Here, c and d are empirical coefficients which are obtained from the zero-order moment. In our study using moment analysis, we evaluated m_0 values to check if the probability was preserved. If the

```
#=============================================================================================
#%% import modules
import numpy as np

# Degradation term
def Z(t, c, d):
        """
    Parameters
    ----------
    t  : Time at point of measurement
    c,d: Empirical coefficents from zero order moment

    Returns
    -------
    Degradation at time t
    """
    return d * np.exp (c*t)
#=============================================================================================
```

Fig. 3. Code for degradation term.

```
#=============================================================================================
#%% Import modules
import numpy as np

# Drift term using moment analysis
def drift_term_moment(rm, sig):
     """
    Parameters
    -----------
    sig: sigma
    rm: median pore radius

    Returns
    -------
    Drift term (first order normalized moment)
    """
    return rm * np.exp((sig**2)/2)
#=============================================================================================
# Drift term using existing mathematical expression
def drift_term_expression(t, a, b, rm):
     """
    Parameters
    -----------
    rm: median pore radius
    a : temporal value of drift term
    b : absolute value of drift term
    t : Time at point of measurement

    Returns
    -------
    Drift term from existing mathematical expression
    """
    r = (b * rm)/(rm + (b - rm)*np.exp(-a * t))
    return (a * (1-(r/b)) * r)

#=============================================================================================
# Cumulative drift term
def CT(f, a, b, N) :
      """
    Parameters
    -----------
    f  : Integrand (drift term function)
    a,b: initial and final number of days of measurmeents
    N  : number of time steps

    Returns
    -------
    Cumulative drift term
    """
       t = np.linspace(a, b, N)
       ft = f(t)
       CuT = np.sum(ft) * (b-a)/N
       return (CuT)
#=============================================================================================
```

Fig. 4. Code for drift term.

probability was not preserved, we included the degradation term in the analytical solution. The code for the degradation term is shown in Fig. 3.

Drift term

The first-order normalized moment M_1 characterizes the mean pore size $\langle r \rangle$ [L]:

$$M_1 = \quad < r > \quad = \quad r_m \quad \exp\left(\frac{\sigma^2}{2}\right) \tag{9}$$

The first-order moment can also be obtained from other independent models [1]. For instance, the rather popular expression from Thornley [10] is used for the drift term.

$$V(t) = \frac{d}{dt}\langle r\rangle = a\left(1 - \frac{\langle r\rangle}{b}\right)\langle r\rangle \;\; where \;\; \langle r\rangle = \frac{b\langle r_0\rangle}{\langle r_0\rangle + (b - \langle r_0\rangle)\exp(-at)} \tag{10}$$

The code for both these scenarios are shown in Fig. 4.

Dispersion term

Finally, the second-order centralized moment μ_2 characterizes the variance [11]:

$$\mu_2 = r_m^2 \exp(\sigma^2)\left[\exp(\sigma^2) - 1\right] \tag{11}$$

However, our lack of knowledge on how the dispersion behaves with respect to the pore size paved the way to use the dispersivity λ.

$$\lambda = \frac{D(t)}{|V(t|} \tag{12}$$

λ is obtained by fitting Eq. (5) to the observed values by means of the Levenberg-Marquardt method

a

```
#=============================================================================================================================
#%% import modules
import numpy as np
from scipy import integrate

# Fraction term with the exponential
M = integrate.quad(Z, 0, T)[0]

# Exponential term before the intergral
E = np.exp(M/T) # Exponential term

# Optimization of lambda
def f0(x):
    """
    Parameters
    ----------
    TSi: initial measured saturated water content
    TRi: initial measured residual water content
    sigi: initial sigma
    rmi: initial median pore radius

    Returns
    -------
    Initial pore size distribution
    """
        return (TSi - TRi)/(x * sigi* np.sqrt(2*np.pi)) * (np.exp((-(np.log(x/rmi))**2)/(2*Si**2)))

def part1(x,a,b,lam):
        """
    Parameters
    ----------
    a,b: integration variables
    lam: lambda, the parameter to be optimized

    Returns
    -------
    Part of the analytical solution
    """
        return ((1/(np.sqrt(4*b*lam*np.pi))  * ((np.exp(-((a-x+b)**2)/(4*b*lam))) + (np.exp((-a/lam) -
(((a+x-b)**2)/(4*b*lam)))))) + ((1/(2*lam)) * np.exp(-a/lam) * special.erfc((a+x-b)/(np.sqrt(4*b*lam))))

def part2(x,a,b,lam):
     """
    Parameters
    ----------
    a,b: integration variables
    lam: lambda, the parameter to be optimized

    Returns
    -------
    Pieced together parts of the analytical solution
    """
        return f0(x) * (part1(x,a,b,lam))
#=============================================================================================================================
```

Fig. 5. a) Part 1 of the code for optimizing lambda (and obtaining the analytical solution). b) Part 2 of the code for optimizing lambda (and obtaining the analytical solution).

b
```
#=============================================================================================================================
# Analytical solution
def I(a,b,lam):
    """
    Parameters
    ----------
    a,b: integration variables
    lam: lambda, the parameter to be optimized

    Returns
    -------
    Analytical solution (without the exponential part)
    """
        II = integrate.quad(part2, 0, np.inf, args = (a,b,lam))[0]
        return II

# Predicted values
def pred(r_meas,lam):
    """
    Parameters
    ----------
    r_meas: Pore radius at which water retention parameters were measured
    lam: lambda, the parameter to be optimized

    Returns
    -------
    Predictions for the optimization process
    """
     PSD=[]
     for i in range(len(r_meas)):
         PS = I(r_meas[i], T, lam) # T is the cumulative drift term
         PSD.append(PS)
     return PSD
#=============================================================================================================================
# Observed values to optimize lambda
# PSD is already defined in Fig. 1
# o stands for observed
fit = PSD(r_meas, TSo, TRo, sigo, rmo)
popt, pcov = curve_fit(pred, r_meas, fit)   # Levenberg-Marquardt optimization
# popt gives the optimized value of lambda
#=============================================================================================================================
```

Fig. 5. (Continued)

Analytical solution

As a final step, the pore size distribution (analytical solution in Eq. 5) is obtained using the code in Fig. 5a and b.

The optimized lambda value should now be used in the code (Fig. 5a and b) instead of lam to get the predicted pore size distribution curves. The reader is directed to Chandrasekhar et al. [1] for the output curves.

Outlook

The possibility of using other independent models for the coefficients exists and is being looked into. Further, the optimization process for lambda necessitates the final water retention parameter values making the model redundant. However, lack of sufficient data sets that capture the temporal dynamics of soil structure hampers our quest to establish a range of values for the coefficients as well as for the validation process. In conclusion, the model is able to capture very well the evolution of soil pore size distribution in its current state with scope for improvement in how we estimate the coefficients.

Acknowledgements

This work was supported by the German Research Foundation (DFG) (grant number SCHW 1448/6-1,FE 504/11-1) and the Austrian Science Fund (FWF) (grant number I-2122 B16). The authors would also like to thank Dr. Arun Kannawadi for his support with the Python code.

Appendix A. Supplementary data

Supplementary material related to this article can be found, in the online version, at doi:https://doi.org/10.1016/j.mex.2019.09.014.

References

[1] P. Chandrasekhar, J. Kreiselmeier, A. Schwen, T. Weninger, S. Julich, K.-H. Feger, K. Schwärzel, Modeling the evolution of soil structural pore space in agricultural soils following tillage, Geoderma 353 (2019) 401–414.
[2] P. Chandrasekhar, J. Kreiselmeier, A. Schwen, T. Weninger, S. Julich, K.-H. Feger, K. Schwärzel, Why we should include soil structural dynamics of agricultural soils in hydrological models, Water 10 (2018) 1862.
[3] H.-J. Vogel, S. Bartke, K. Daedlow, K. Helming, I. Kögel-Knabner, B. Lang, E. Rabot, D. Russell, B. Stößel, U. Weller, et al., A systemic approach for modeling soil functions, Soil 4 (2018) 83–92.
[4] D. Or, F.J. Leij, V. Snyder, T.A. Ghezzehei, Stochastic model for posttillage soil pore space evolution, Water Resour. Res. 36 (2000) 1641–1652.
[5] F.J. Leij, T.A. Ghezzehei, D. Or, Modeling the dynamics of the soil pore-size distribution, Soil Tillage Res. 64 (2002) 61–78.
[6] F.J. Leij, T.A. Ghezzehei, D. Or, Analytical models for soil pore-size distribution after tillage, Soil Sci. Soc. Am. J. 66 (2002) 1104–1114.
[7] K. Kosugi, Three-parameter lognormal distribution model for soil water retention, Water Resour. Res. 30 (1994) 891–901.
[8] M.T. Van Genuchten, A closed-form equation for predicting the hydraulic conductivity of unsaturated soils, Soil Sci. Soc. Am. J. 44 (1980) 892–898.
[9] K. Schwärzel, S. Carrick, A. Wahren, K.-H. Feger, G. Bodner, G. Buchan, Soil hydraulic properties of recently tilled soil under cropping rotation compared with two-year pasture, Vadose Zone J. 10 (2011) 354.
[10] J.H.M. Thornley, A new formulation of the logistic growth equation and its application to leaf area growth, Ann. Bot. 66 (1990) 309–311.
[11] J. Aitchison, J.A.C. Brown, The Lognormal Distribution, Cambridge University Press, Cambridge, 1963.