main   NoahPy / README.md

wbtian Update README.md   6574437 · 6 minutes ago

183 lines (136 loc) · 9.8 KB

Preview   Code   Blame   Raw

NoahPy

A new version of the modified Noah land surface model (Noah LSM v3.4.1) with full backpropagation support. This model has been re-coded in Pytorch to be fully differentiable, and it includes several key improvements to its representation of physical processes for freeze-thaw cycles compared to the orignal Noah LSM v3.4.1. Detailed descriptions of these enhancements, along with their applications in simulating permafrost thermal–hydrological processes, can be found in the following references:

- Zhang G, Nan Z, Hu N, Yin Z, Zhao L, Cheng G, Mu C. Qinghai-Tibet Plateau permafrost at risk in the late 21st century. Earth's Future. 2022, 10(6): e2022EF002652. https://doi.org/10.1029/2022EF002652

- Wu X, Nan Z, Zhao S, et al. Spatial modeling of permafrost distribution and properties on the Qinghai-Tibet Plateau. Permafrost and Periglacial Processes, 2018, 29(2): 86–99. https://doi.org/10.1002/ppp.1971

- Chen H, Nan Z, Zhao L, et al. Noah modelling of the permafrost distribution and characteristics in the West Kunlun Area, Qinghai-Tibet Plateau, China. Permafrost and Periglacial Processes, 2015, 26(2): 160–174. https://doi.org/10.1002/ppp.1841

A technical documentation, including a brief introduction to the theoretical basis, implementation details, and performance evaluation, is also included with the code for further information.

Authors: Wenbiao Tian & Zhuotong Nan (Orcid) (nanzt@njnu.edu.cn)

Web: https://permalab.science

Citation: Tian, W. & Nan, Z. (2025). wbtian/NoahPy: NoahPy v1.0.1. Zenodo. https://doi.org/10.5281/zenodo.16530326

July 31, 2025

Code Structure

This folder contains the NoahPy source code. The parameter_new directory holds essential lookup tables, including those for soil parameters and other preprocessed input data.

Two versions of the NoahPy code are included:

- NoahPy.py: A procedural implementation with all functions written independently.

- NoahPy_module.py: An object-oriented implementation where functions are encapsulated within classes. This version generally offers better execution performance.

Both versions implement the same core processes and are interchangeable for use.

Usage

The usage of NoahPy varies slightly depending on the version you choose:

Using NoahPy_module (object-oriented version)

```python
from NoahPy_Module import NoahPy
model = NoahPy()
Date, STC, SH2O = model.noah_main(file_name, output_flag=False, lstm_model=None)
```

- file_name: Path to your input file.
- output_flag: If set to True, the results will be written to an output file. The default is False.
- lstm_model: An optional argument for coupling with an LSTM model; Set to None if you're not using an LSTM.

Using NoahPy.py (procedural version)

```python
from NoahPy import noah_main
Date, STC, SH2O = noah_main(file_name, output_flag=False, lstm_model=None)
```

The input parameters and output format are identical to those in the object-oriented version.

Input Requirements

- The input forcing file must adhere to the structure used in the Noah LSM. Please refer to the provided **forcing.txt** file for an example.

- The parameter_new folder contains essential model lookup tables:

  - VEGPARM.TBL: Vegetation parameter table

  - SOILPARM.TBL: Soil parameter table

- General model parameters (originally from GENPARM.TBL) have been hardcoded into the model for simplicity. You may modify these directly in the source code if needed.

Main Processes
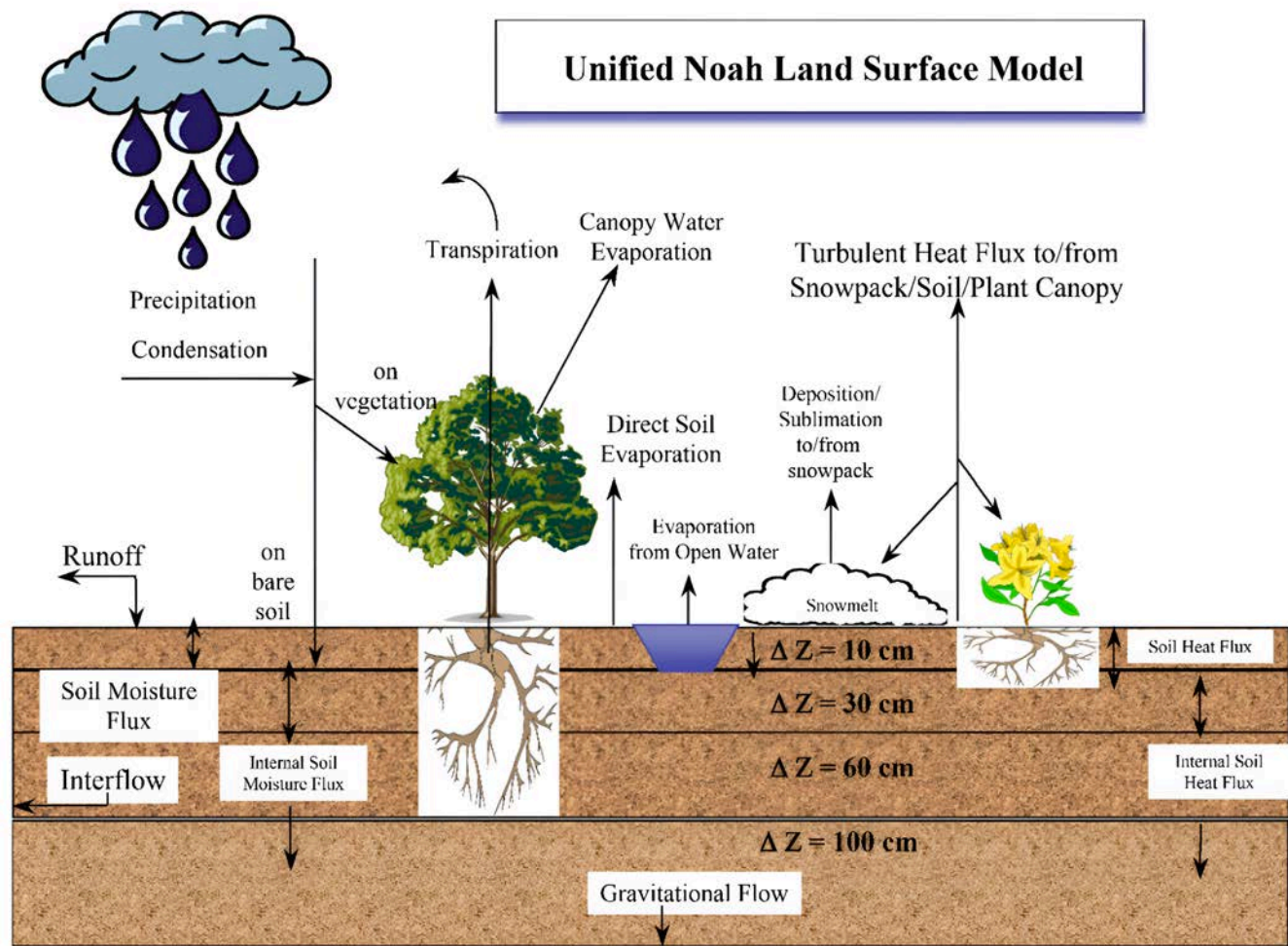
RNN-wrapped main processes
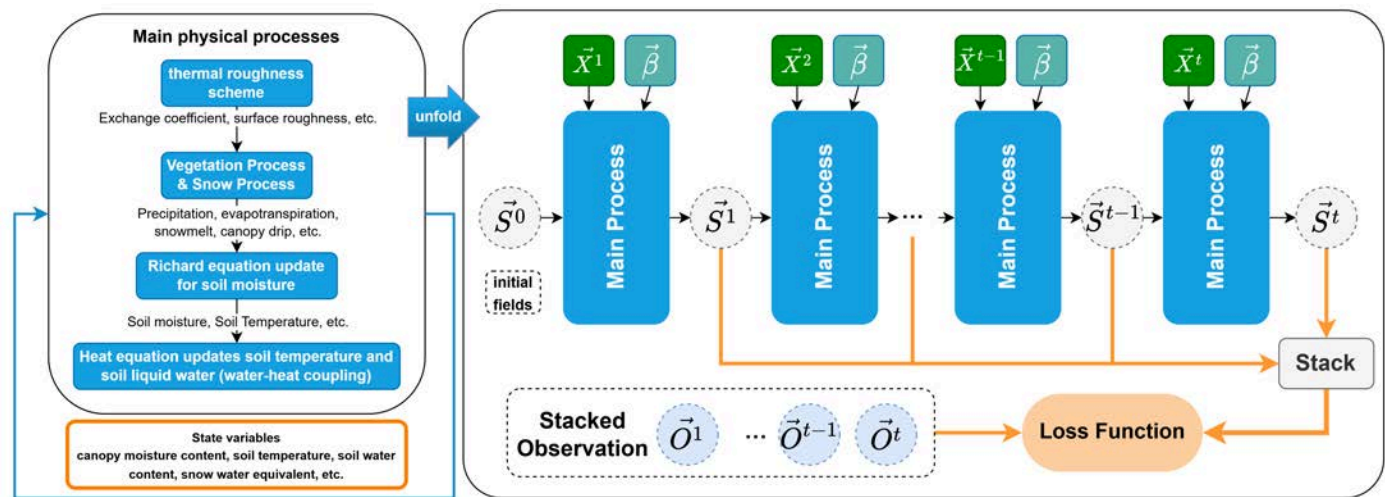
This diagram illustrates the RNN-wrapped physical processes architecture, where $\vec{S}$, $\vec{X}$, $\vec{O}$ represent the meteorological forcing, state, and observation vectors, respectively, $\vec{\beta}$ represents the model parameter vector.

Basic equation

In the land surface process model, the two most fundamental laws are the heat equation and soil water movement equation, both of which are partial differential equations. These equations are often solved using finite difference methods, which ultimately convert them into a system of equations. This conversion allows for the use of differentiable solution methods from machine learning platforms (such as PyTorch and TensorFlow) to enable gradient propagation within the model.

(1) Heat equation

$$\frac{\partial}{\partial t}\rho C_p T = \frac{\partial}{\partial z}[\frac{\partial KT}{\partial z}] + Q$$

where, T_s represents the soil temperature (K); C_s represents the soil heat capacity (J·m^{-3}·K^{-1}), λ is the soil thermal conductivity (W·m^{-1}·K^{-1}). The calculation process for C_s is as follows:

$$C_s = \theta C_w + (1 - \theta_s) C_{soil} + (\theta_s - \theta) C_{air}$$

where, θ represents the soil water content (m^{-3}·m^{-3}); s indicates the soil porosity (m^{-3}·m^{-3}); C_w, C_{soil}, and C_{air} represent the heat capacity of water, soil substrate, and air (J·m^{-3}·K^{-1}), respectively.

The fomular for calculating soil heat conduction (λ) is:

$$\lambda(\theta) = K_e (\lambda_{sat} - \lambda_{dry}) + \lambda_{dry}$$

where K_e is Kersten number, λ_{sat} is the thermal conductivity of saturated soil (W·m^{-1}·K^{-1}), and λ_{dry} is the thermal conductivity of dry soil (W·m^{-1}·K^{-1}).

(2) Richards' equation

NoahPy describes soil water movement using the Richards equation, formulated as follows:

$$\frac{\partial \theta}{\partial t} = \frac{\partial}{\partial z}[D(\theta)\frac{\partial \theta}{\partial z}] + \frac{\partial K(\theta)}{\partial z} + S$$

where: θ represents the saturated volumetric water content (m^{-3}·m^{-3}); t stands for time (s); D is the soil-water diffusivity (m^2·s^{-1}); K is the hydraulic conductivity (m·s^{-1}), Ψ is the soil matric potential (m), z is the soil depth (m); S represents soil water sources and sinks (e.g., infiltration, evapotranspiration). In this formula, the first term to the right of the equal sign represents the part of soil moisture diffusion driven by the gradient of the soil vertical water potential Ψ. The second term on the right side of the equation indicates the part of soil moisture conduction caused by gravity.

In NoahPy, the soil hydaulic conductivity K and soil matrix potential Ψ are calculated using the Campbell parameterization scheme:

$$K(\theta) = K_s (\theta / \theta_s)^{2b+3}$$

$$\Psi(\theta) = \Psi_s (\theta / \theta_s)^{-b}$$

where: Ψ_s is the soil water potential at air entry (m); K_s and θ_s represent the saturated hydraulic conductivity (m·s^{-1}) and saturated volumetric water content (m^{-3}·m^{-3}), respectively. b is an empirical parameter (dimensionless) related to the pore size distribution of the soil matrix.

Finite Difference Discretization
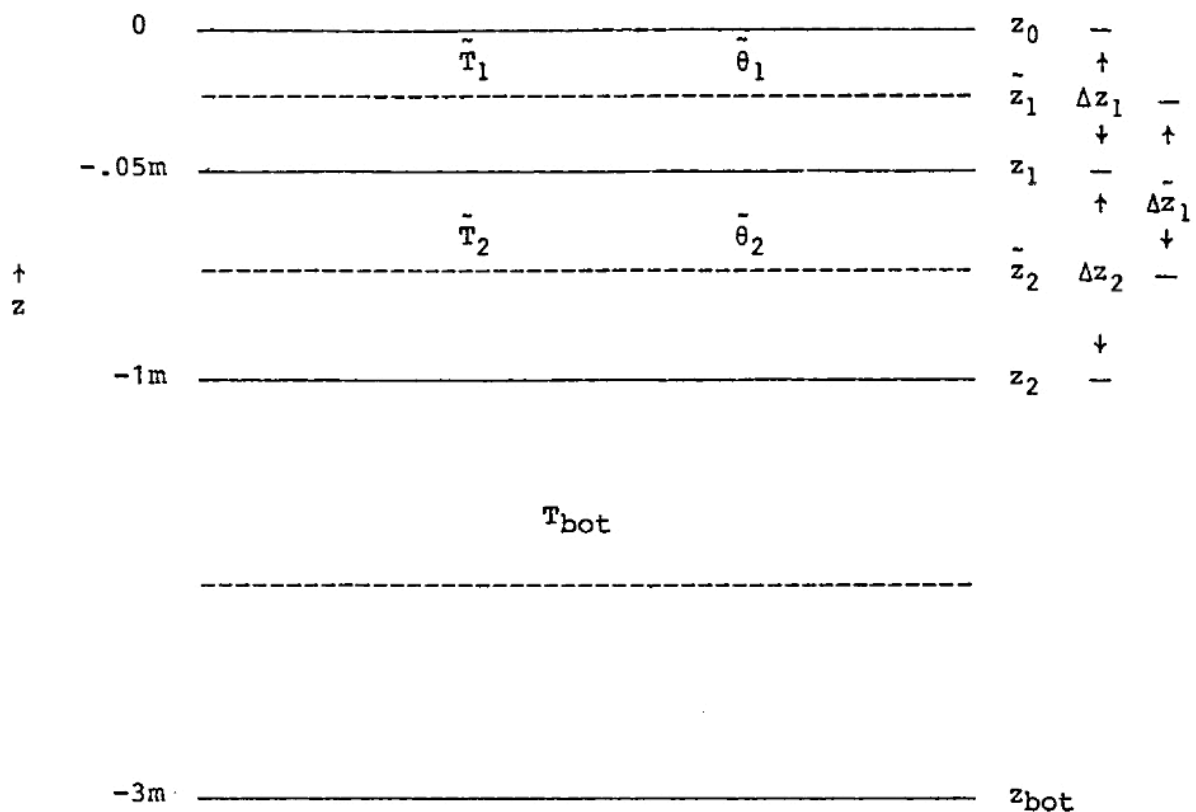
Soil discrete schematic diagram

Fig. A-1.   The geometry of the soil thermodynamics.

- Pan H L, Mahrt L. Interaction between soil hydrology and boundary-layer development[J]. Boundary-Layer Meteorology, 1987, 38(1-2): 185-202. doi: 10.1007/BF00121563.
- Kalnay E, Kanamitsu M. Time Schemes for Strongly Nonlinear Damping Equations[J]. Monthly Weather Review, 1988, 116(10): 1945-1958. doi: 10.1175/1520-0493(1988)116<1945:TSFSND>2.0.CO;2.

The finite difference form of the equation can be derived using the above discretization scheme and the time scheme "D" (IMPLICIT STATE, EXPLICIT COEFFICIENT) as presented in Section 2 of Kalnay & Kanamitsu:

We obtain:

$$\frac{\theta_k^{t+1} - \theta_k^t}{\Delta t} = \frac{1}{\Delta z_k}[D(\theta_{k-1})\frac{\theta_{k-1}^{t+1} - \theta_k^{t+1}}{\Delta \tilde{z}_{k-1}} - D(\theta_k)\frac{\theta_k^{t+1} - \theta_{k+1}^{t+1}}{\Delta \tilde{z}_k} + K_{k-1} - K_k + S]$$

$$\frac{\theta_k^{t+1} - \theta_k^t}{\Delta t} = -\frac{D(\theta_{k-1})}{\Delta z_k \Delta \tilde{z}_{k-1}}(\theta_k^{t+1} - \theta_{k-1}^{t+1}) - \frac{D(\theta_k)}{\Delta z_k \Delta \tilde{z}_k}(\theta_k^{t+1} - \theta_{k+1}^{t+1}) + \frac{S + K_{k-1} - K_k}{\Delta z_k}$$

Let:

$$A = -\frac{D(\theta_{k-1})\Delta t}{\Delta z_k \Delta \tilde{z}_{k-1}}, C = -\frac{D(\theta_k)\Delta t}{\Delta z_k \Delta \tilde{z}_k}$$

Finally:

$$A(\theta_{k-1}^{t+1} - \theta_{k-1}^t) + B(\theta_k^{t+1} - \theta_k^t) + C(\theta_{k+1}^{t+1} - \theta_{k+1}^t) = RHS$$

where,

$$RHS = [\frac{S + K_{k-1} - K_k}{\Delta z_k} \cdot \Delta t + A(\theta_k^t - \theta_{k-1}^t) + C(\theta_k^t - \theta_{k+1}^t)], \ B = 1 - (A + C)$$

Using matrix representation:

$$
\begin{bmatrix}
B_1 & C_1 & 0 & 0 & 0 & \cdots & 0 \\
A_2 & B_2 & C_2 & 0 & 0 & \cdots & 0 \\
0 & A_3 & B_3 & C_3 & 0 & \cdots & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \cdots & 0 \\
0 & \cdots & 0 & 0 & A_{k-1} & B_{k-1} & C_{k-1} \\
0 & \cdots & 0 & 0 & 0 & A_k & B_k
\end{bmatrix}
\begin{bmatrix}
\theta_1^{t+1} - \theta_1^t \\
\theta_2^{t+1} - \theta_2^t \\
\theta_3^{t+1} - \theta_3^t \\
\vdots \\
\theta_{k-1}^{t+1} - \theta_{k-1}^t \\
\theta_k^{t+1} - \theta_k^t
\end{bmatrix}
=
\begin{bmatrix}
RHS_1 \\
RHS_2 \\
RHS_3 \\
\vdots \\
RHS_{k-1} \\
RHS_k
\end{bmatrix}
$$

This simplified to: $PX = D$

This linear equation system, $PX = D$, can then be solved using the differentiable method available in the machine learning platforms.