



## User Manual

# **TMELand: Quantitative modeling and visualization of Waddington's epigenetic landscape and state transition paths**

Lin Zhu<sup>1,†</sup>, Xin Kang<sup>2,3,†</sup>, Pei Lin<sup>1</sup>, Chunhe Li<sup>2,3,4,\*</sup> and Jie Zheng<sup>1,\*</sup>

<sup>1</sup>School of Information Science and Technology, ShanghaiTech University, Shanghai, 201210, China, <sup>2</sup>School of Mathematical Sciences, Fudan University, Shanghai, 200433, China, <sup>3</sup>Shanghai Center for Mathematical Sciences, Fudan University, Shanghai, 200433, China and <sup>4</sup>Institute of Science and Technology for Brain-Inspired Intelligence, Fudan University, Shanghai, 200433, China

Availability: <https://github.com/JieZheng-ShanghaiTech/TMELand>

Contact: [zhengjie@shanghaitech.edu.cn](mailto:zhengjie@shanghaitech.edu.cn), [chunheli@fudan.edu.cn](mailto:chunheli@fudan.edu.cn)

## Contents

1. Introduction.....	3
1.1. Waddington's epigenetic landscape and extension.....	3
1.2. Related work and motivation.....	3
1.3. An overview of the TMELand.....	3
2. Installation.....	4
2.1. Dependency package summary.....	4
2.2. Installation pipeline.....	5
3. Model definition.....	5
3.1. Loading files.....	5
3.1.1. TSV.....	6
3.1.2. XPPAUT ODE.....	6
3.1.3. SBML.....	7
3.1.4. TME.....	7
3.2. Model visualization.....	9
3.3. Model parameters definition.....	11
4. Landscape construction and visualization.....	12
4.1. Trajectories simulation.....	12
4.2. Probabilistic landscape.....	13
4.3. Visualization.....	14
4.4. Save landscape.....	15
5. Path generation and drawing.....	15
5.1. Transition path.....	15
5.2. Visualization.....	16
5.3. Save path.....	17
6. Case studies.....	17
6.1. Case study 1: Cancer attractors.....	17
6.2. Case study 2: Stem cell differentiation and reprogramming.....	18
6.3. Case study 3: EMT-metabolism network.....	20
7. Time and memory test.....	21
8. License.....	23
9. Contact information.....	23
Reference.....	23

# 1. Introduction

## 1.1. Waddington’s epigenetic landscape and extension

Waddington’s epigenetic landscape was proposed by C. H. Waddington in 1957 for the first time, which is a metaphor to describe the cellular developmental process [1]. In the original Waddington’s theory, the process of a marble rolling down a hill represents the process of differentiating into a specific type of cell. Wang et al extend the concept to study reverse differentiation processes such as cell reprogramming [2].

## 1.2. Related work and motivation

For the construction of Waddington’s landscape, there are two mainstream approaches: data-driven and model-driven. [3] and [4] proposed data-driven methods to construct the landscapes by learning Hopfield network from gene expression data. Meanwhile, model-driven methods include probability-based methods [5, 6], Boolean network-based methods [7], and a Monte Carlo method [8].

Here, we focus on the model-driven methods for landscape modeling. Recently, a few software tools have been developed to model and visualize Waddington’s landscape conveniently, including NetLand [9] and ATLANTIS [10]. NetLand is another desktop application to visualize Waddington’s landscape based on ODE described GRNs, which is developed by Java. ATLANTIS is a MATLAB toolbox to visualize and analyze landscapes based on Boolean networks. The transition path between attractors of a landscape is a key feature to research reprogramming. However, NetLand doesn’t have the functionality of finding transition path, and the landscapes constructed are not accurate enough.

## 1.3. An overview of the TMELand

We develop a lightweight, python-based application TMELand for differential equations based gene regulatory network models to achieve visualized analysis.

TMELand can load three main kinds of models, TSV, ODE computation model, and self-defined model (TME) for saving and reloading. TSV text file can reflect the topology of the gene regulatory network (GRN). ODE computation model includes

XPPAUT ODE [11] and SBML two types. XPPAUT ODE refers to the .ode file, it includes a set of ode equations and some parameters. SBML is based on the XML file format, which stores computational models of biological processes. We can download SBML model files from the BioModels Database [12], where the BioModels Database contains rich mathematical models of biological systems. After loading the model, TMELand can visualize models, i.e., it can transform a TSV text file into a GRN topology network and parsed the ODE computation model into ODE equations. Then you can specify a set of parameters to construct a landscape, you can also adjust visualization-related parameters to obtain a pretty landscape. When you finish construction of landscape, you can draw paths between attractors by specify beginning and ending attractors.

TMELand is developed by python and can be used in many platforms, including Windows, Unix, and macOS. The detailed reliable packages are introduced in the 2.1 section.

## 2. Installation

### 2.1. Dependency package summary

This is a summary of the dependency package in our test environment. Our python version is 3.7.0.

**Table 1.** The dependency-package summary of TMELand desktop application.

Name	Version	Summary	License
<b>matplotlib</b>	3.1.2	Python plotting package.	PSF
<b>networkx</b>	2.5	Python package for creating and manipulating graphs and networks.	BSD
<b>numpy</b>	1.18.1	NumPy is the fundamental package for array computing with Python.	BSD
<b>scipy</b>	1.5.2	Scientific Library for Python.	BSD
<b>python-libsmbml</b>	5.18.0	LibSBML is a library for reading, writing, and manipulating the Systems Biology Markup Language (SBML).	LGPL
<b>sympy</b>	1.7.1	SymPy is an open-source Python	BSD

## 2.2. Installation pipeline

The pipeline is commonly used for all platforms. You can open the command line in the TMELand folder, and input the following commands. (We recommend using conda to create a new virtual environment and then install packages.)

1. Create a virtual environment and install dependency packages

```
conda create -n TMELand_env python==3.7
conda activate TMELand_env
pip install -r requirements.txt
```

2. Launch the TMELand

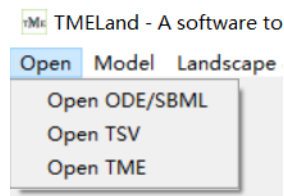
For Windows and Ubuntu:	<code>python ./MainPage.py</code>
For macOS:	<code>python ./MainPageForMac.py</code>

**Figure 1.** The pipeline of install and launch TMELand.

## 3. Model definition

### 3.1. Loading files

TMELand can load three main kinds of models, TSV, ODE computation model, and self-defined model using JSON format. Specifically, the ODE computation model contains the XPPAUT ODE format file and SBML format file.



**Figure 2.** The ‘Open’ entry in the menu bar.

When choosing ‘Open ODE/SBML’, you can change the file type to show the ODE or SBML file.

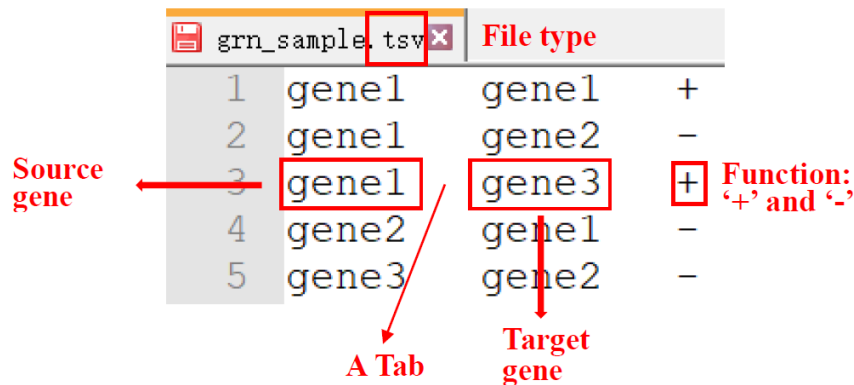


**Figure 3.** Choosing file type when choosing ‘Open ODE/SBML’ entry.

### 3.1.1. TSV

TSV (Tab-separated values) format file stores a collection of interaction relationships of genes with a specific definition, which can reflect the gene regulatory network topology.

Each line of a TSV file includes two gene names and one interaction relationship between these two genes, ‘+’ represents the first gene activates the second gene, and ‘-’ represents the first gene inhibits the second gene. Each item in a line is separated by a tab. An example with instructions is given as follow:



	Source gene	Target gene	Function
1	gene1	gene1	+
2	gene1	gene2	-
3	gene1	gene3	+
4	gene2	gene1	-
5	gene3	gene2	-

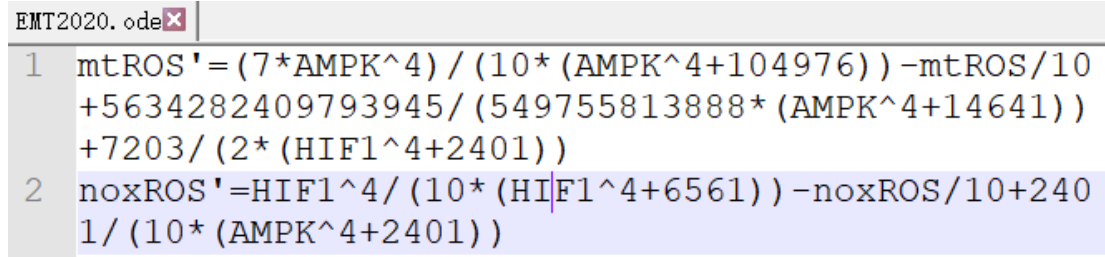
**Figure 4.** A TSV input file example and instructions.

### 3.1.2. XPPAUT ODE

XPPAUT ODE format file consists of a set of ODE equations and correlated parameters. In our support ODE models, you can follow the following rules. [13] (XPPAUT ODE format file parsing code refer to <https://github.com/MCLand-NTU/MCLand>)

1. (Optional) Ignore lines that begin with ‘#’ or ‘@’;
2. (Optional) Define parameters by using a line starting with ‘**par**’, multiple parameters are separated by a comma;
3. (Momentary) Define differential equations by starting with ‘**Var**=’ or ‘**dVar/dt**=’, **Var** refers to a variable;
4. (Optional) Define constant by starting with ‘const’;
5. (Optional) Define initial values by starting with ‘init’;

This is an example:



```

EMT2020.ode
1 mtROS'=(7*AMPK^4)/(10*(AMPK^4+104976))-mtROS/10
+5634282409793945/(549755813888*(AMPK^4+14641))
+7203/(2*(HIF1^4+2401))
2 noxROS'=HIF1^4/(10*(HIF1^4+6561))-noxROS/10+240
1/(10*(AMPK^4+2401))

```

**Figure 5.** An example of part of an XPPAUT ODE format file.

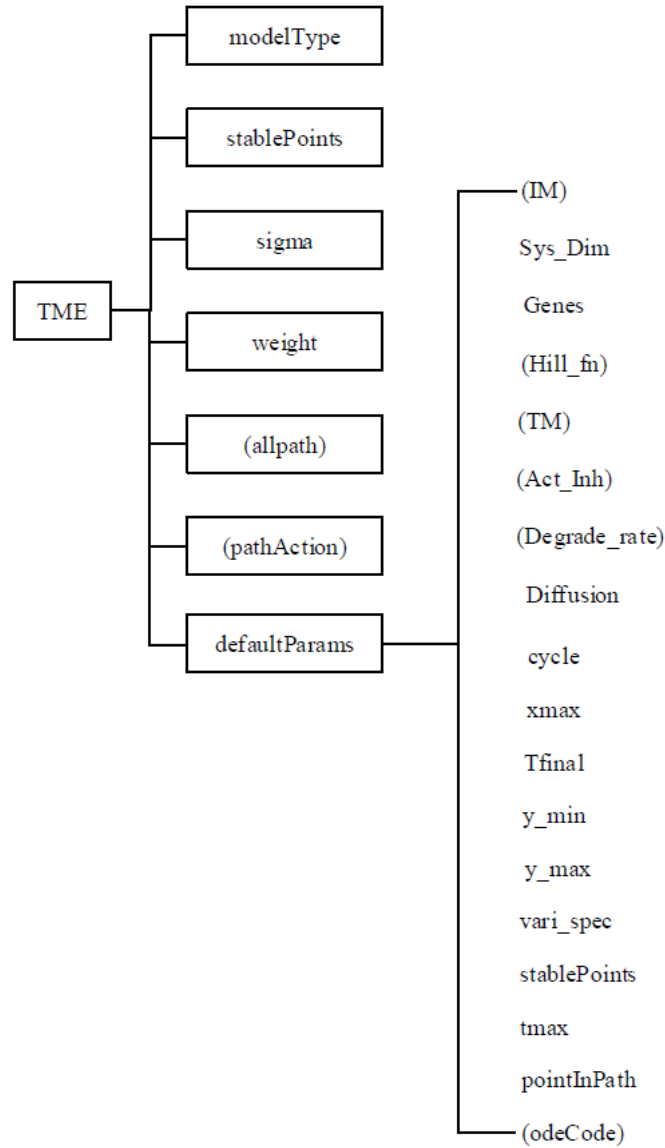
### 3.1.3. SBML

SBML is an abbreviation of Systems Biology Markup Language. It is a representation format, based on XML, for communicating and storing computational models of biological processes [14]. Generally, we can use an SBML file that describes transcriptional regulations from the BioModels Database [12]. BioModels Database contains rich mathematical models of biological systems with a specific format. For more information, please refer to <https://www.ebi.ac.uk/biomodels/>.

Due to the complication of SBML, we don't show an example here. You can refer to the 'ODE\_SBML\_models' folder in TMEland code for a full example.

### 3.1.4. TME

TME is a self-defined model using JSON format, which is used to save computed model and next reloading. We can save landscape only or landscape with paths. This is the structure of a TME JSON file ('allPath' and 'pathAction' only appear when calculated paths; IM, 'Hill\_fn', TM, 'Act\_Inh', 'Degrade\_rate' are defined for TSV models; 'odeCode' are defined for ODE based models mainly):



**Figure 6.** The structure of the TME file (JSON type).

We will introduce the 7 main items, the detailed items of default\_params are given in the later sections. modelType denotes the format of the input file (TSV file: ‘GRN’; ODE/SBML file: ‘ODE’; TME file: append ‘TME’ after the loaded model); stablePoints are converged locations of all trajectories; sigma is covariance; weight can reflect the width of basin (stablePoints, sigma, and weight are key elements to construct landscape); allpath stores all paths that you have drawn with dictionary format, the key is the index of beginning attractor to the index of end attractor, and the value is variation of all genes along the path; pathAction represents the difficulty level of transfer, the

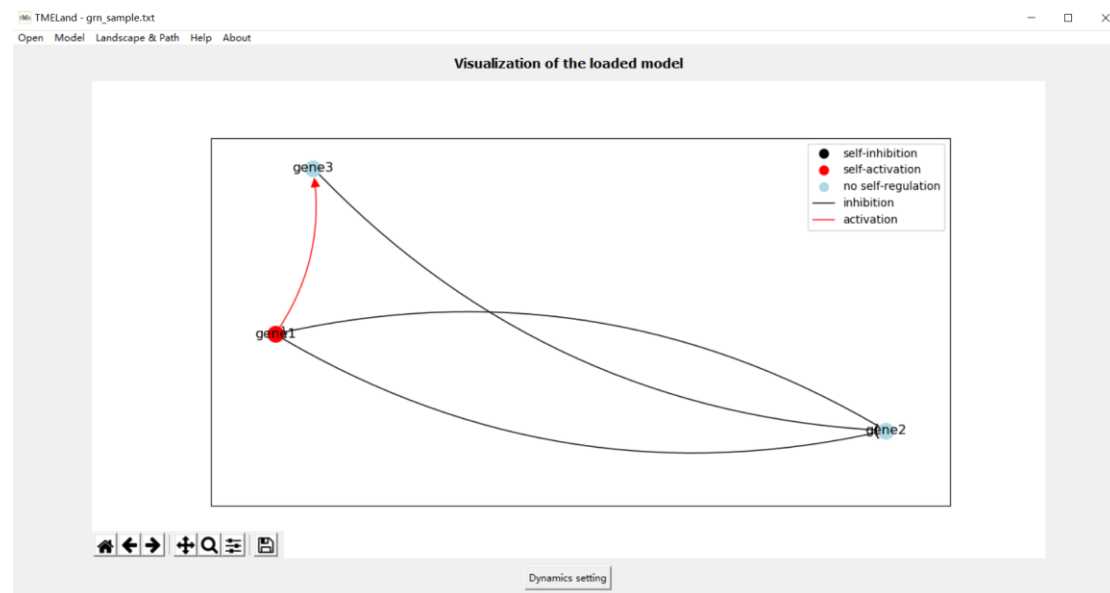


higher of pathAction is, the harder to transfer; default\_params contains defined parameters in the TMELand and other useful information.

### 3.2. Model visualization

After loaded models, we provide the visualization functionality.

For the TSV model, it would be visualized as a network with arrows. Nodes represent genes and different-color arrows represent different interacted relationships. Red node means the gene is self-activation, black node means the gene is self-inhibition, light blue node means the gene doesn't have a self-regulation relationship. The red line with an arrow represents activation and the black line with a bar represents inhibition.



**Figure 7.** An example of a TSV model visualization.

For XPPAUT ODE or SBML models, they are parsed as a set of differential equations and displayed in the TMELand.

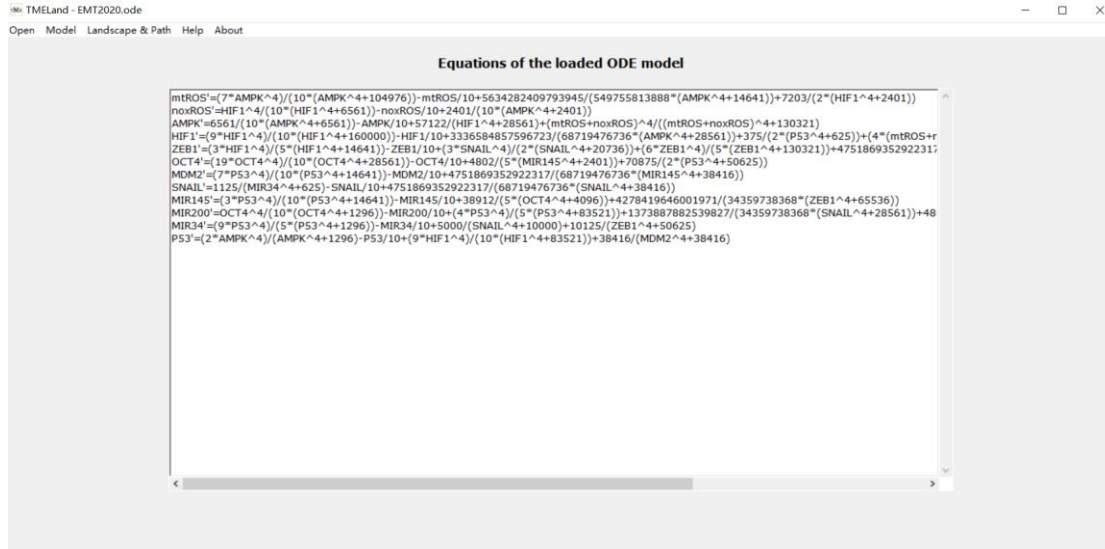


Figure 8. An example of XPPAUT ODE model visualization.

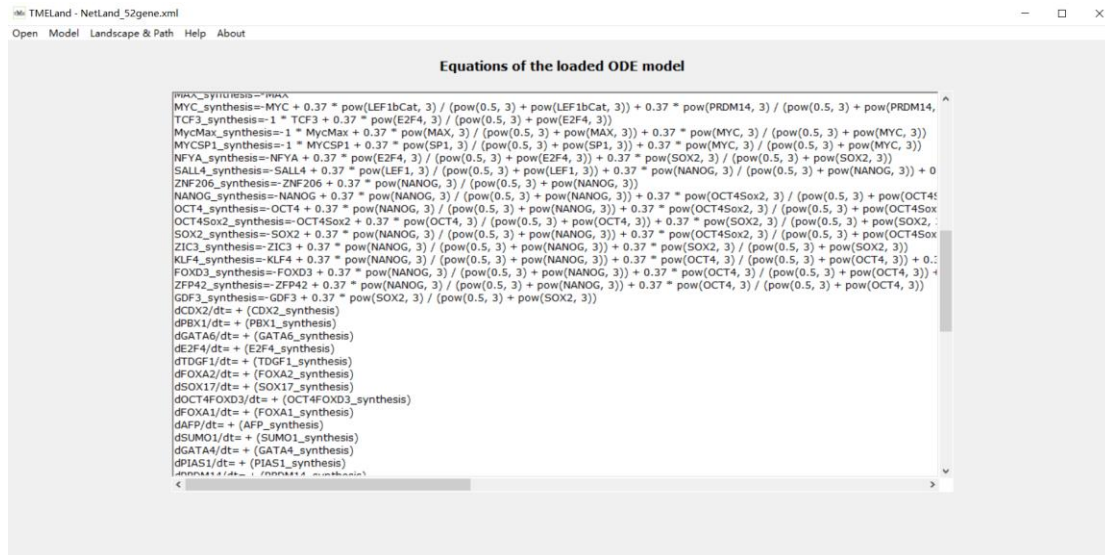
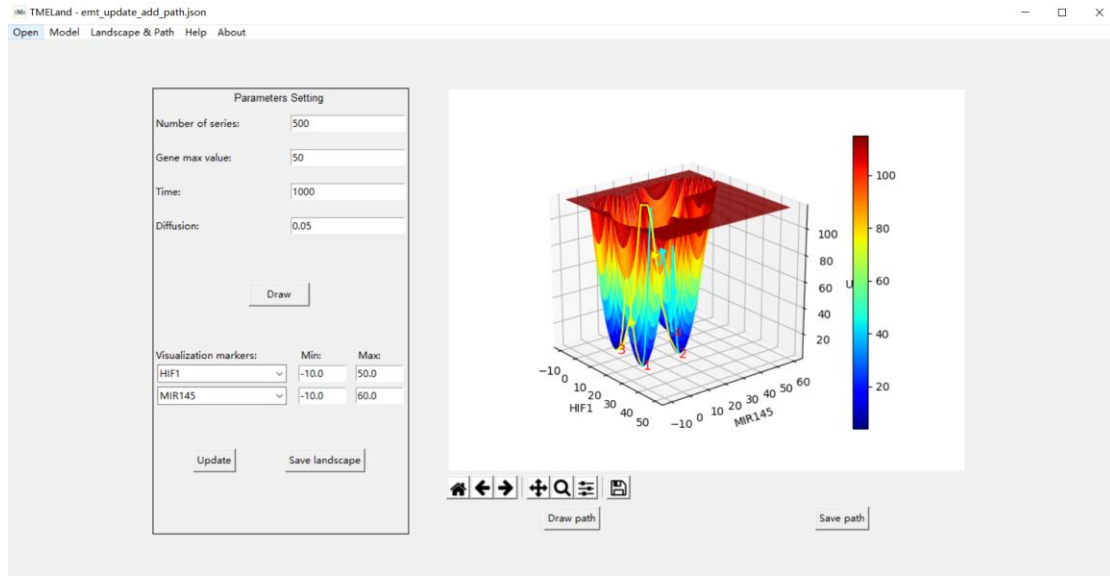


Figure 9. An example of SBML [model](#) visualization [9].

For TME models, they are visualized on the Landscape page in the TMELand.



**Figure 10.** An example of TME model visualization.

### 3.3. Model parameters definition

TMELand is based on the differential equations, thus, we need to define parameters to form differential equations.

Obviously, we only need to transform the TSV model into differential equations, because we can use ODE-based models (XPPAUT ODE and SBML) to construct landscape directly.

We use the modeling method proposed in [6] to construct differential equations of GRN topology. Specifically, we need to define parameters in Equation (1) to form the ODE equation for each gene. In Equation (1),  $x_i$  is the expression level of the  $i$ th gene,  $m_{i1}$  is the number of activations to gene  $i$ ,  $m_{i2}$  is the number of inhibitions to gene  $i$ ,  $x_{pji}$  is activation gene for gene  $i$ ,  $x_{qji}$  is the inhibition gene for gene  $i$ ,  $A$  is activation constant,  $B$  is inhibition constant,  $n$  is Hill coefficient,  $s$  is the threshold, and  $k$  is degradation rate.

$$\frac{dx_i}{dt} = \sum_{j=1}^{m_{i1}} \frac{Ax_{pji}^n}{x_{pji}^n + s^n} + \sum_{j=1}^{m_{i2}} \frac{Bs^n}{x_{qji}^n + s^n} - kx_i \quad (1)$$

Correspondingly, after we visualize the GRN topology, we will go into the ‘Dynamics setting’ page to define these parameters, and the interface is shown as follow:

Show GRN

**Dynamics (ODE) related parameters setting**

Hill coefficient: 4

Threshold: 0.7

Activation: 1

Inhibition: 0.5

Degradation: 1

ODE generation

**Figure 11.** The interface of the ‘Dynamics setting’ page.

You can click ‘ODE generation’ to generate the set of differential equations corresponding to all genes.

## 4. Landscape construction and visualization

### 4.1. Trajectories simulation

After we obtain differential equations of GRN by loading the input file, we simulate multiple trajectories to analyze the steady states of the system. In the ‘Landscape’ page, we define three parameters related to simulation, ‘Number of series’, ‘Gene max value’, and ‘Time’. ‘Number of series’ refers to the number of trajectories. One trajectory corresponding to an initial condition, and the initial gene expression value is randomly generated under the constraint of maximum gene expression value (‘Gene max value’). ‘Time’ is the initial evolution time points of the differential equation. Time can be set large when ODE converges slowly, and vice versa.

Number of series:	500
Gene max value:	1
Time:	1000

**Figure 12.** The screenshot of parameters definition related to simulation.

#### 4.2. Probabilistic landscape

TMELand uses the probability-based method to visualize Waddington's landscape, which applies the truncated moment equation (TME) method proposed in [6]. With the development of all simulated trajectories, we will acquire the steady-state points  $\bar{\mathbf{x}}$  finally when trajectories are converged, and converged points are also known as basins of attractors. The ratio of terminal points in each basin is its weight  $\mathbf{w}$ . Then, by using the calculated steady-state points  $\bar{\mathbf{x}}$  and self-defined diffusion coefficient, we can calculate the covariance  $\boldsymbol{\sigma}$ . We will determine the shape of the landscape using the weighted sum of multiple normal distributions  $P_{ss}$  with mean  $\bar{\mathbf{x}}$ , covariance  $\boldsymbol{\sigma}$ , and weight  $\mathbf{w}$ . Once we have the steady-state probability distribution  $P_{ss}$ , we calculate the potential  $U$  of landscape, i.e.,  $U = -\ln(P_{ss})$  [5, 6].

We define the diffusion coefficient parameter after three simulation-related parameters. Then, we click the 'Draw' button to visualize the constructed landscape with default visualization parameters (the default visualization parameters are using the first two genes as marker genes and setting each gene range is from 0 to 3).

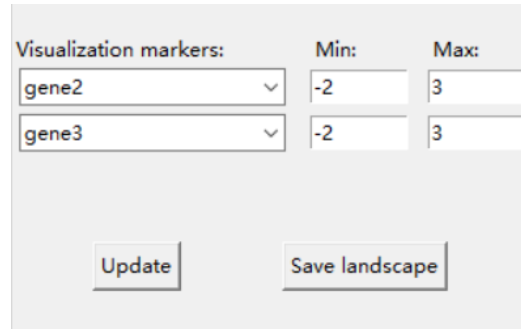
Parameters Setting	
Number of series:	500
Gene max value:	1
Time:	1000
Diffusion:	0.005

Draw

**Figure 13.** The screenshot of parameters to draw a landscape.

### 4.3. Visualization

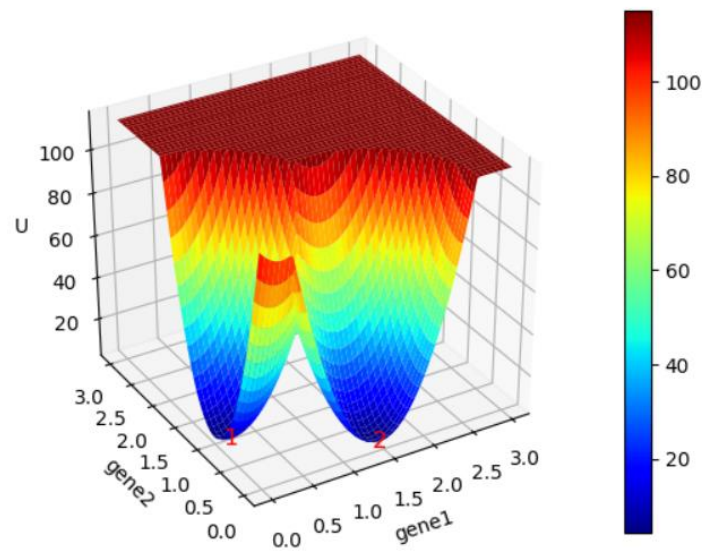
To visualize the landscape, we select two marker genes as the x-axis and y-axis respectively and define the value range of axes. From the last subsection, we know that using default visualization parameters to visualization by ‘Draw’ button. Then, we can choose different marker genes and ranges to update landscape by the button ‘Update’.



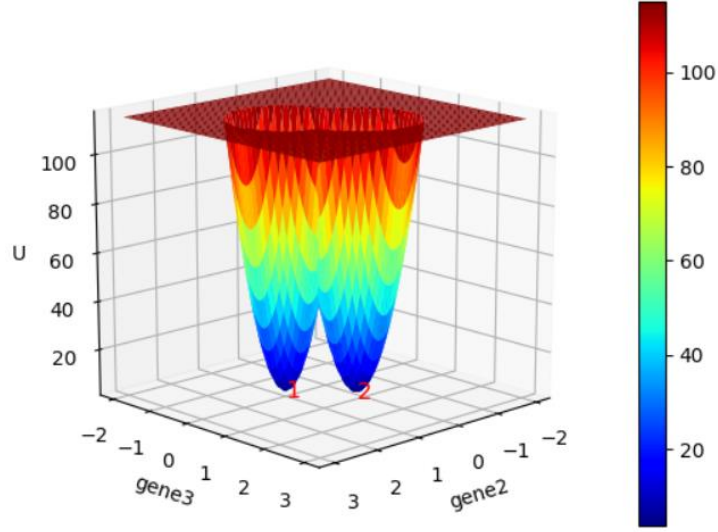
The screenshot shows a user interface for updating visualization parameters. It features a section titled 'Visualization markers:' with two dropdown menus. The first dropdown is set to 'gene2' and the second to 'gene3'. To the right of these are two columns labeled 'Min:' and 'Max:'. For 'gene2', the 'Min:' value is '-2' and the 'Max:' value is '3'. For 'gene3', the 'Min:' value is '-2' and the 'Max:' value is '3'. At the bottom of the interface are two buttons: 'Update' and 'Save landscape'.

**Figure 14.** The screenshot of parameters to update landscape.

Then, we show a visualized landscape (grn\_sample.tsv model in TSV\_models folder) corresponding to the configuration of Fig. 13 and Fig. 14 separately.



**Figure 15.** The ‘Draw’ result of configuration in Fig. 13.



**Figure 16.** The ‘Update’ result of configuration in Fig. 14.

We can update the landscape many times until we acquire a pretty one. After the update, if we need to change parameters before the ‘Draw’ button (i.e., model and simulation-related parameters), we can click the ‘Draw’ button again with new parameters and the last time visualization parameter. Furthermore, if we load a TME model, we can also change visualization-related parameters to update the landscape.

#### 4.4. Save landscape

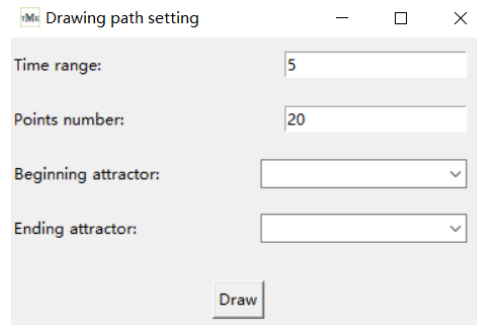
We can save the computed landscape for the next reloading by clicking the ‘Save landscape’ button. The save format is JSON, we call the TME model here. For the detailed definition of the TME model please refer to the [TME](#) section.

## 5. Path generation and drawing

### 5.1. Transition path

After drawing the landscape, a more important task is finding the transition paths between attractors. The quantification of kinetic transition paths among attractors provides guidance for us to understand cellular differentiation and reprogramming [5]. TMELand provides the function by solving the optimization problem to obtain the minimum action paths (MAP) as transition paths [15]. You can draw a path by clicking

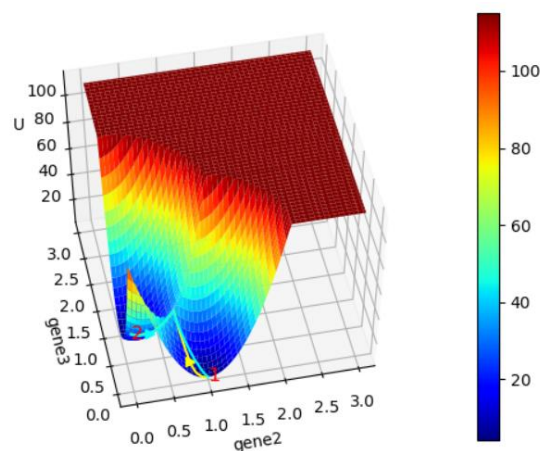
the ‘Draw path’ button on the Landscape page after finish the construction of a landscape. Then, we can define the optimal problem of path-related parameters (i.e., time range and granularity) and specify the beginning and ending attractor to draw the transition path.



**Figure 17.** The screenshot of parameters setting to draw a path.

## 5.2. Visualization

Similar to constructed landscape, to visualize paths, we should reduce dimension from the high dimensional multiple genes to two marker genes. In this way, we obtain the 2D coordinates of paths, and then we project them onto the landscape with the same potential as the landscape. Specifically, we mesh the x-y plane into 100x100 grids, and project points of the path into grids. Every grid has a z value of the landscape projected into this grid. We can acquire z values of all points in the corresponding grid. In this way, we can draw a path in the 3D space with 3D coordinates (x, y, and z). To denote a path more clearly, we add a triangle on the path to represent the direction and the triangle is near the source point.



**Figure 18.** An example of drawing paths on a landscape.



It's worth noticing that path from attractor 1 to 2 is not consistent with the path from attractor 2 to 1, which may denote a different process. The variation of the marker gene expression value along transition paths may give us insights into the corresponding biological process.

Furthermore, if we load a TME model, we can go on drawing paths directly. This allows us to divide the whole process into two separate sub-processes. We can also update the landscape with paths using the 'Update' button.

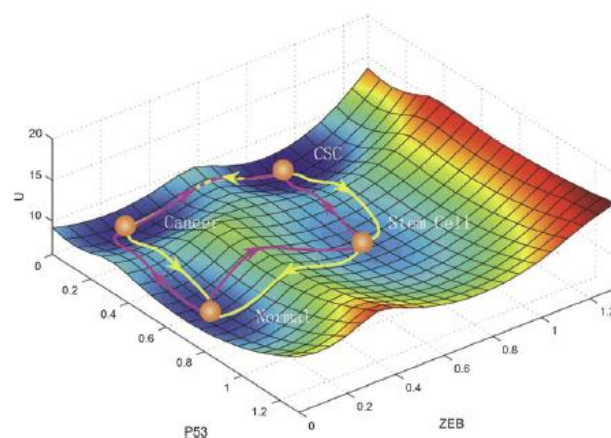
### 5.3. Save path

We can save the model after drawing transition paths by clicking the 'Save path' button. The save format is JSON, we call the TME model here. For the detailed definition of the TME model please refer to the [TME](#) section.

## 6. Case studies

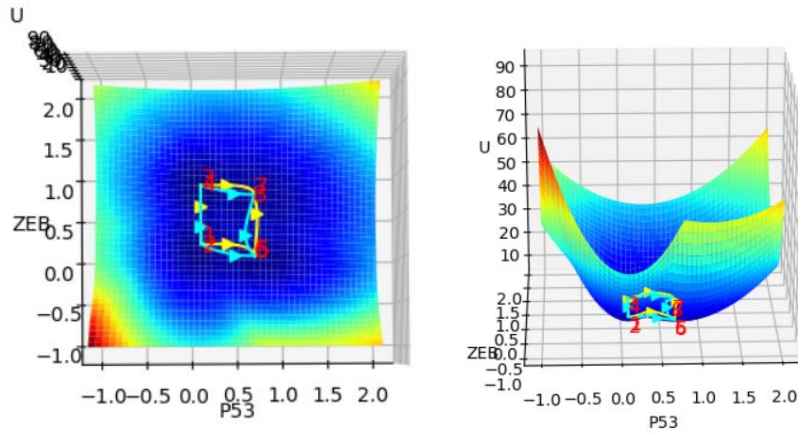
### 6.1. Case study 1: Cancer attractors

For case study 1, we use the 6-gene model proposed in [16]. The model is developed to research the development of cancer stem cells (CSC). In their original work, they calculated 4 attractors, CSC, cancer, normal, and stem cell. Additionally, they also calculated transition paths among attractors (Fig. 19).



**Figure 19.** The calculated landscape and transition paths in the original work of Case study 1 [16].

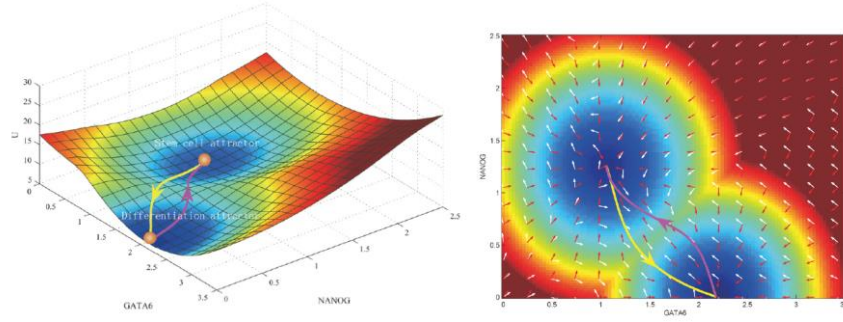
We use TMELand to load the XPPAUT ODE format model (./CaseStudies/CaseStudy1\_Li6\_ODE.ode) and draw landscape and transition paths as shown in Fig. 20. From Fig. 20, we obtain 8 attractors, and attractors 1, 2 can seem like a pair, 3 and 4, 5 and 6, 7 and 8 likewise. We draw paths among attractors 1, 3, 5, 7 and we find that paths are consistent with the original work. Accordingly, attractor 3, 4 represents CSC, attractor 1, 2 represents cancer, attractor 7, 8 represents stem cell, and attractor 5, 6 represents normal. Furthermore, we both test NetLand and MATLAB version MCLand [8], they all give out 8 attractors, which means we may give more details than the original work. The phenomenon is especially mentioned in [8]. The computed model is saved as CaseStudy1\_Li6\_Result.json, you can load the TME file by using TMELand to reproduce Case study 1.



**Figure 20.** The result of Case study 1 using TMELand.

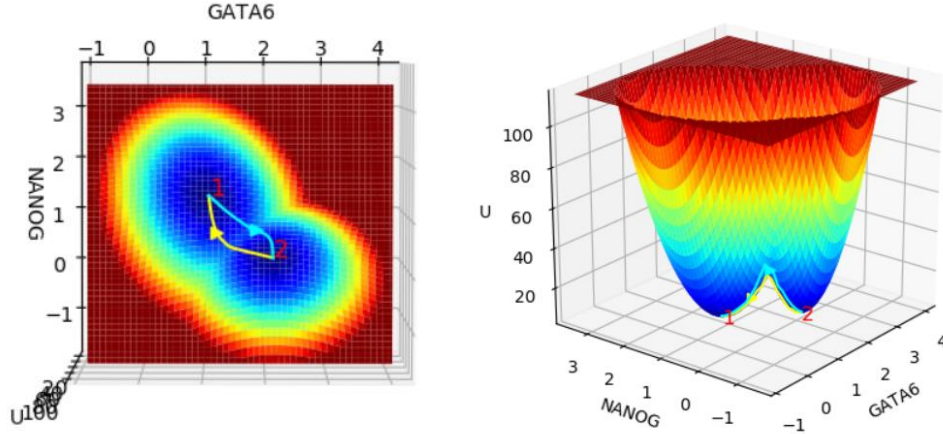
## 6.2. Case study 2: Stem cell differentiation and reprogramming

For case study 2, we use the 52-gene model proposed in [5]. The model is developed to research the process of differentiation and reprogramming. In their original work, they calculated 2 attractors, one represents stem cell attractor and another one represents differentiation attractor. They also calculated transition paths between these two attractors to find out how the differentiation and reprogramming happened (Fig. 21).



**Figure 21.** The calculated landscape and transition paths in the original work of Case study 2 [5].

We use TMELand to load the XPPAUT ODE format model (`./CaseStudies/CaseStudy2_Li52_ODE.ode`) and draw landscape and transition paths as shown in Fig. 22. From Fig. 22, we obtain 2 attractors. By comparing the results generated by TMELand and results in their original work, we find that they are consistent. Accordingly, attractor 1 represents stem cell attractor, and attractor 2 represents differentiation attractor. The computed model is saved as `CaseStudy2_Li52_Result.json`, you can load the TME file by using TMELand to reproduce Case study 2.

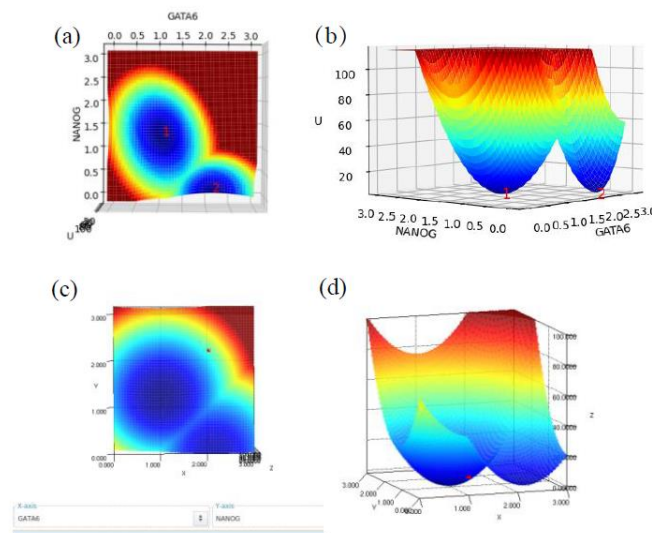


**Figure 22.** The result of Case study 2 using TMELand.

We test NetLand on Case study 2 and using the same parameters to compare the results of TMELand and NetLand. We set the ‘Number of series’ as 100, ‘Gene max value’ is 1, ‘Time’ is 128, and the visualization range is from 0 to 3. The results of TMELand are (a), (b) in Fig. 23 and the results of NetLand are (c), (d) in Fig. 23. The test file is `./CaseStudies/NetLand_52gene.xml` [9].

In the paper of the TME method, they have pointed out that they consider correlations

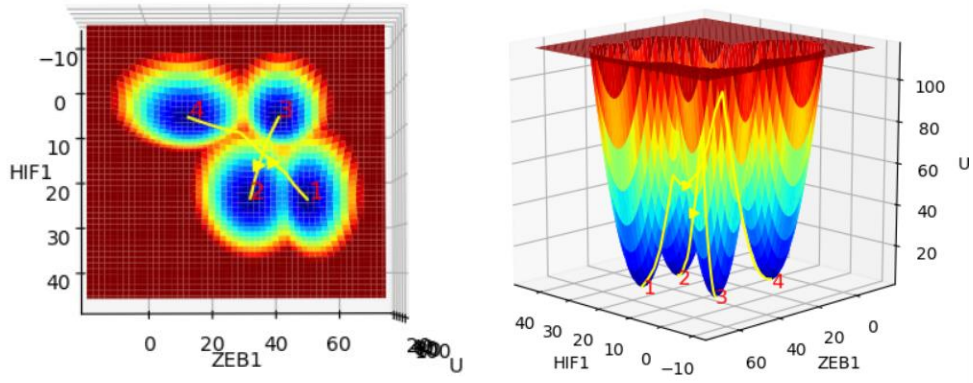
between any two genes, which means genes are not independent in GRN [6]. Take all correlations of genes into account is also more meaningful. As we can see in Fig. 23, the axis of attractor in (a) is oblique, which means GATA6 has a relationship with NANOG. However, we can't find a strong correlation of marker genes in (c). Thus, the comparison implies that TMELand is much better.



**Figure 23.** The comparison of TMELand and NetLand for Case study 2 using the same parameters. (a) and (b) are results of TMELand; (c) and (d) are results of NetLand.

### 6.3. Case study 3: EMT-metabolism network

The EMT-metabolism network has the same topology as the EMT-metabolism subnetwork of the GRN in [6]. In [6], they construct a GRN composed of metabolism, EMT, and metastasis to characterize cancer. However, the EMT-metabolism network here has different models and parameters, which obtain four stable points. We use TMELand to load the XPPAUT ODE model (./CaseStudies/CaseStudy3\_EMT2020\_ODE.ode) and draw landscape and transition paths as shown in Fig. 24. The computed model is saved as CaseStudy3\_EMT2020\_Result.json, you can load the TME file by using TMELand to reproduce Case study 3.



**Figure 24.** The result of Case study 3 using TMELand.

## 7. Time and memory test

We test the performance of TMELand on all case studies (XPPAUT ODE models), and the number of genes is 6, 12, 52. The test environment is a 12-core CPU 3.70 GHz Intel(R) Core(TM) i7-8700K Ubuntu 18.04.2 LTS 64-bit system and 32G memory. The version of Python is 3.7.0. We use the time and tracemalloc package to measure the consuming time and memory separately.

We compare time-consuming and memory costs for both computing landscape and transition path. The consuming time for landscape and path is separate, and the memory is all peak value when the process is running. The computing parameters are described in Table 2.

**Table 2.** The parameters setting for performance test.

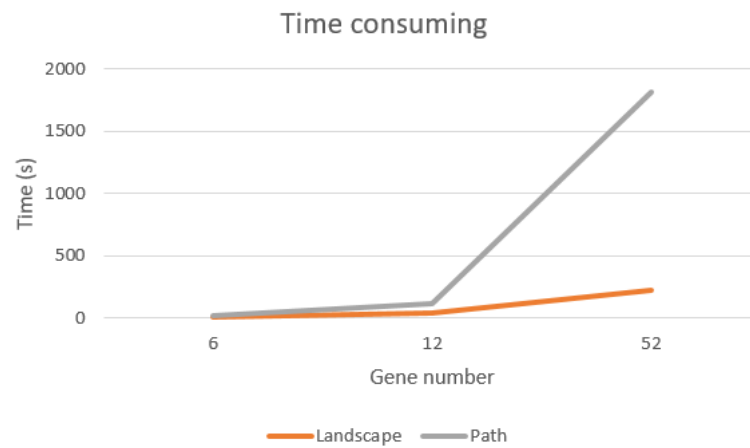
	Case study 1 (6)	Case study 2 (52)	Case study 3 (12)
Number of series	500	500	500
Gene max value	1	1	50
Time	1000	1000	1000
Diffusion	0.005	0.005	0.05
Markers	p53, Zeb	NANOG, GATA6	HIF1, ZEB1
Visualization range	[-2,3], [-2,3]	[-1.5,3], [-0.5,4]	[-10,45], [-15,70]
Time range	5	5	5

Points number	20	20	20
Beginning attractor	1	1	1
Ending attractor	2	2	2

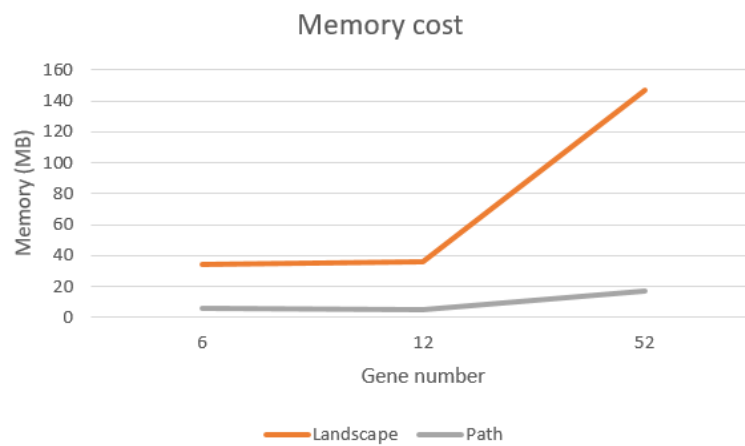
The comparison result as follows:

**Table 3.** The comparison of time and memory cost of TMELand for different N (N represents gene number).

		N=6	N=12	N=52
<b>Time (s)</b>	<b>Landscape</b>	11.2	36.9	224.1
	<b>Path</b>	19.9	114.6	1815.6
<b>Memory (MB)</b>	<b>Landscape</b>	34.7	36.0	146.7
	<b>Path</b>	5.8	4.9	16.9



**Figure 25.** The time-consuming changes with an increase in the number of genes for test models.



**Figure 26.** The memory cost changes with the increase in the number of genes for test models.

From Fig. 25 and Fig. 26, we can watch that with the increase of the number of genes, both time-consuming and memory cost increase gradually. However, as for time-consuming, the time of computing a path grows faster than computing the landscape. The memory cost of the landscape is much higher and grows faster than a path. Furthermore, we also noticed that different model types have different performances. For example, we both compare the performance of XPPAUT ODE format and TSV format of N=52 model, and the latter is much slower than the former. That may because of many for-loops in computing for a TSV model, such as the process of forming ODEs for a TSV model. So, it's better to provide an ODE-based model, rather than a TSV model when you pursue computing efficiency.

## 8. License

TMELand followed the GNU General Public License (GPL) v3.0 license and as found in the LICENSE file in the main directory. If you have any problem, please contact [zhengjie@shanghaitech.edu.cn](mailto:zhengjie@shanghaitech.edu.cn) or [chunheli@fudan.edu.cn](mailto:chunheli@fudan.edu.cn).

## 9. Contact information

TMELand has been thoroughly tested, you can use it freely with instructions of this User Manual. However, we can't cover all models during our test. If there is any bug, error, or improvement when you use it, please post your issue on the GitHub website (<https://github.com/JieZheng-ShanghaiTech/TMELand>) or send your suggestions or bug reports to [zhengjie@shanghaitech.edu.cn](mailto:zhengjie@shanghaitech.edu.cn) or [chunheli@fudan.edu.cn](mailto:chunheli@fudan.edu.cn).

## Reference

1. Waddington, C.H., *The strategy of the genes*. 2014: Routledge.
2. Wang, J., et al., *Quantifying the Waddington landscape and biological paths for development and differentiation*. 2011. **108**(20): p. 8257-8262.
3. Maetschke, S.R. and M.A.J.B. Ragan, *Characterizing cancer subtypes as attractors of*



- Hopfield networks*. 2014. **30**(9): p. 1273-1279.
4. Guo, J. and J.J.B. Zheng, *HopLand: single-cell pseudotime recovery using continuous Hopfield network-based modeling of Waddington's epigenetic landscape*. 2017. **33**(14): p. i102-i109.
  5. Li, C. and J.J.P.C.B. Wang, *Quantifying cell fate decisions for differentiation and reprogramming of a human stem cell network: landscape and biological paths*. 2013. **9**(8): p. e1003165.
  6. Kang, X., J. Wang, and C.J.i. Li, *Exposing the underlying relationship of cancer metastasis to metabolism and epithelial-mesenchymal transitions*. 2019. **21**: p. 754-772.
  7. Flöttmann, M., T. Scharp, and E.J.F.i.p. Klipp, *A stochastic model of epigenetic dynamics in somatic cell reprogramming*. 2012. **3**: p. 216.
  8. Zhang, X., et al., *A Monte Carlo method for in silico modeling and visualization of Waddington's epigenetic landscape with intermediate details*. 2020. **198**: p. 104275.
  9. Guo, J., et al., *NetLand: quantitative modeling and visualization of Waddington's epigenetic landscape using probabilistic potential*. 2017. **33**(10): p. 1583-1585.
  10. Shah, O.S., et al., *ATLANTIS-Attractor landscape analysis toolbox for cell fate discovery and reprogramming*. 2018. **8**(1): p. 1-11.
  11. Ermentrout, B., *Simulating, analyzing, and animating dynamical systems: a guide to XPPAUT for researchers and students*. 2002: SIAM.
  12. Le Novère, N., et al., *BioModels Database: a free, centralized database of curated, published, quantitative kinetic models of biochemical and cellular systems*. 2006. **34**(suppl\_1): p. D689-D691.
  13. Ket Hing Chong, X.Z. and J.Z., *MCLand: A Python program for drawing spontaneously emerging paths in Waddington's epigenetic landscape by Monte Carlo estimation*. Submitted, 2021.
  14. Wikipedia. *SBML*. 2020; Available from: <https://en.wikipedia.org/wiki/SBML>.
  15. Weinan, E., et al., *Minimum action method for the study of rare events*. 2004. **57**(5): p. 637-656.
  16. Li, C. and J.J.C.r. Wang, *Quantifying the landscape for development and cancer from a core cancer stem cell circuit*. 2015. **75**(13): p. 2607-2618.