proto (2016/4/12)

index
/home/shotaro/Dropbox/Workspace/6/lab/growing-string/proto.py

```
Numerical simulation of growing strings
```

各点間をバネでつなぎ、その自然長が時間と共に増大することを考える。
この自然長、もしくは2点間の距離がある閾値を超えた時、新たに2点間に点を置く。
点に作用する力は、バネによる力と、曲げ弾性による力、粘性による摩擦力である。
オイラー法または4次のルンゲクッタ法でこれを解き、matplotlibでアニメーションにして表示する。

Modules

matplotlib.animation	numpy	random
logging	matplotlib.pyplot	time

Classes

__builtin__.object

- Euler
- RK4

Points
String_Simulation

class **Euler**(__builtin__.object)

Methods defined here:

__init__(self, function)
```
Initialize function.
```

solve(self, y, t, h)
```
Solve the system ODEs.

--- arguments ---
y: Array of initial values (ndarray)
t: Time (float)
h: Stepsize (float)
```

Data descriptors defined here:

__dict__
```
dictionary for instance variables (if defined)
```

__weakref__
```
list of weak references to the object (if defined)
```

class **Points**

Methods defined here:

__init__(self, N, position_x, position_y, natural_length, K, length_limit)
```
Initialize class variants.

--- Arguments ---
N              (int)    : How many points should placed
position_x     (ndarray): Array of the valuse of x axis for each points
position_y     (ndarray): Array of the valuse of y axis for each points
natural_length (ndarray): Array of natural length of each strings
K              (ndarray): Array of spring constant
length_limit   (float)  : Threshold for dividing to 2 strings
```

create_new_point(self, k, X)
新しい点を2点の間に追加し，各物理量を再設定する

k番目とk+1番目の間に新しい点を追加

divide_if_extended(self, X)
2点間の自然長がlength_limitの設定値より大きいとき新しい点を追加する

get_distances(self, x_list, y_list)
```
Caluculate distance between two points and return list.

--- Arguments ---
x_list (list or ndarray): x座標の値のリスト
```

y_list (list or ndarray): y座標の値のリスト

grow(self, func_nl, func_k)
2点間の自然長を大きくする & バネ定数を変化させる

バネ定数は単位(自然長)長さあたりが同じいなるように随時変化させる
--- Arguments ---
func_nl (function): N-1(開曲線),N(閉曲線)次元のnp.arrayに対する関数
返り値は同次元のnp.arrayで返し，これが成長後の自然長のリストである
func_k (function): N-1(開曲線),N(閉曲線)次元のnp.arrayに対する関数
返り値は同次元のnp.arrayで返し，これが成長後のバネ定数のリスト

update_natural_length(self, k, d)
自然長を更新

Called from self.**create_new_point**
Change: self.**natural_length**

update_point_position(self, k)
点を追加

Called from self.**create_new_point**
Change: self.**position_x**, self.**position_y**

update_point_velocity(self, k)
速度を更新

Called from self.**create_new_point**
Change: self.**vel_x**, self.**vel_y**

update_spring_constant(self, k)
バネ定数を更新

Called from self.**create_new_point**
Change: self.**K**

class **RK4**(__builtin__.object)

Methods defined here:

__init__(self, function)
Initialize function.

solve(self, y, t, h)
Solve the system ODEs.

--- arguments ---
y: Array of initial values (ndarray)
t: Time (float)
h: Stepsize (float)

Data descriptors defined here:

__dict__
dictionary for instance variables (if defined)

__weakref__
list of weak references to the object (if defined)

class **String_Simulation**

Methods defined here:

__init__(self, parameters)
Assign some initial values and parameters

--- Arguments ---
parameters (dict):
key: x, y, nl, K, length_limit, h, t_max, e, debug_mode
See details for each values in Points's documentation.

animate(self, data)
FuncAnimationから呼ぶ。ジェネレータupdateから返された配列を描画する

force(self, t, X)
各点にかかる力を，バネ弾性と曲げ弾性，粘性の効果を入れて計算

1タイムステップでの変化
@return X'
X = [x, y, x', y']
X' = [x', y', f_x/m, f_y/m]

force_with_more_viscosity(self, t, X)

関数forceの変化形。粘性が優位な場合

1 タイムステップでの変化
@return X'
X = [x, y, x', y']
粘性が非常に大きい時，運動方程式
mx'' = f - D v
のx''は殆ど無視できるとして，式を簡単にすると
D v = f
の式を解くことになる。(x''の項は考慮しなくて良くなる)
X' = [f_x/D, f_y/D, dummy. dummy]

onClick(self, event)
matplotlibの描画部分をマウスでクリックすると一時停止

on_key(self, event)
キーを押すことでシミュレーション中に動作

pause_simulation(self)
シミュレーションを一時停止

run(self)

update(self)
時間発展(タイムオーダーは成長よりも短くすること)

各点にかかる力は，それぞれに付いているバネから受ける力の合力。
Runge-Kutta法を用いて運動方程式を解く。
この内部でglow関数を呼ぶ

--- Arguments ---
point (class): 参照するPointクラスを指定する
h (float): シミュレーションの時間発展の刻み
t_max (float): シミュレーションを終了する時間

Data

__author__ = 'Shotaro Fujimoto'
__date__ = '2016/4/12'

Author

Shotaro Fujimoto