runner


```
# -*- coding:utf-8 -*-
#
# written by Shotaro Fujimoto
# 2016-04-22
```

Modules

matplotlib.animation
logging
numpy
matplotlib.pyplot
random
time

Classes

String_Simulation

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

cross_detect(self, x1, x2, x3, x4, y1, y2, y3, y4)
2つの線分の交差判定を行う

@return True/False
線分1: (x1, y1), (x2,y2)
線分2: (x3, y3), (x4,y4)
線分が接する場合にはFalseを返すこととする

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

update_position_self_avoiding(self)