User Guide of AutoModel

August 6, 2014

Contents

1 What is this program?

Synthetic biology has been developed for a decade and is being expected to evolve the academic society because of its standardization and convenience.[1,2] This novel technology has already yielded bunches of applications on the field of pharmacy[1,2] and therapeutics[3] while also showing its potential on energy industry.[5] However, a standardized model is eagerly required to be established because it will facilitate the exploration of synthetic biology. Therefore, we developed the software – AutoModel, which integrates dozens of models of simple devices and stipulates the I/O interface then provides user standardized modeling path with high degree of freedom.

This python script is being expected to help noviciate to construct their model and decide how much experimental data should be collected. In the coming section, we will demonstrate how this software works and how it help wet-lab experiments. Meanwhile, some instance are listed for illumination.

2 How does it work?

Inspired by the primary idea of synthetic biology, the AutoModel separates existed simple device as individual parts and integrates them into the device file which is a plugin for main program. There are hundreds of mathematical expressions in the device file that represents those relationship between input and output. Customers could add, adapt or amputate any functions, parameters and terms to optimally fitting their experimental data and theoretical model.

$$[A]_{aq} + [B]_{aq} = [A * B]_{aq}$$

3 How does this program help iGEMers?

4 Example

4.1 $f(x, a = 2)$

References

[1] Ellis, Tom, Xiao Wang, and James J. Collins. "Diversity-based, model-guided construction of synthetic gene networks with predicted functions." Nature biotechnology 27.5 (2009): 465-471.

[2] Andrianantoandro, Ernesto, et al. "Synthetic biology: new engineering rules for an emerging discipline." *Molecular systems biology* 2.1 (2006).

[3] Lu, Timothy K., and James J. Collins. "Engineered bacteriophage targeting gene networks as adjuvants for antibiotic therapy." Proceedings of the National Academy of Sciences 106.12 (2009): 4629-4634.

[4] Models for synthetic biology

[5] Lee, Sung Kuk, et al. "Metabolic engineering of microorganisms for biofuels production: from bugs to synthetic biology to fuels." Current opinion in biotechnology 19.6 (2008): 556-563.

5 Appendix

Example #1:

```
# -*- coding: utf-8 -*-
"""
@author: youbin
"""

from pylab import *
from DEVICE_2 import *
import sys

def Initial():
    print "Does the input keep at constant?"

def grow(input, dt = 0.1):
    k1 = 1  ## 1/maximum density
    growrate = 0.5
    output_1 = input + growrate*input*(1 - k1*input)*dt
    return output_1

def Calculate(Tmax):

    dt=0.1
    T = arange(0,Tmax,dt)
    x = zeros_like(T)
    y = zeros_like(T)

    input = 1
    lenth = len(sys.argv)
    sub1 = zeros(lenth)
    x[0] = 0.02
    n = 0
    sub1[0] = 1


    for t in T:

        od = grow(x[n])
        x[n+1] = od

        if lenth >1:
            for i in range(1, lenth):
                #print i
                dev = eval(sys.argv[i])
                #print dev.__doc__
                [sub1[i-1], sub1[i]] = dev(od, sub1[i-1], sub1[i])
            y[n+1] = sub1[lenth-1]

        if n >= Tmax/dt - 2:
            break
        else:
            n = n+1
    return [T,x,y]

if __name__=="__main__":
    #Initial()
```

```
    [T,x_1,x_2] = Calculate(40)
    figure(1)
    xlabel('time')
    ylabel('a.u.')
    ##title('Amplitude of final stage vs. £\mu£')
    plot(T,x_1,label='Density')
    plot(T,x_2,label='output_1')
    legend()
    show()
```

Device File:

```
# -*- coding: utf-8 -*-
"""
@author: youbin
"""

def device_1(od, input, output, dt = 0.1):
    ''' This device is device_1'''  ## Here is the discription of device
    k1 = 0.3
    k2 = 0.5
    k3 = 0
    input_1 = input - k1*od*input*dt
    output_1 = output + (k2*input*od - k3*od)*dt
    return (input_1, output_1)

def device_2(od, input, output, dt = 0.1):
    ''' This device is device_2'''
    k1 = 0.3
    k2 = 0.5
    k3 = 0
    input_1 = input - k1*od*input*dt
    output_1 = output + (k2*input*od - k3*od)*dt
    return (input_1, output_1)

def device_3(od, input, output, dt = 0.1):
    k1 = 0.3
    k2 = 0.5
    k3 = 0
    input_1 = input - k1*od*input*dt
    output_1 = output + (k2*input*od - k3*od)*dt
    return (input_1, output_1)

def device_4(od, input, output, dt = 0.1):
    k1 = 0.3
    k2 = 0.5
    k3 = 0
    input_1 = input - k1*od*input*dt
    output_1 = output + (k2*input*od - k3*od)*dt
    return (input_1, output_1)

def device_5(od, input, output, dt = 0.1):
    k1 = 0.3
    k2 = 0.5
    k3 = 0
    input_1 = input - k1*od*input*dt
    output_1 = output + (k2*input*od - k3*od)*dt
    return (input_1, output_1)
```