RSD_simulator.py

From: Schaffter, SW and Strychalski, EA. *Co-transcriptional RNA strand displacement circuits* Science Advances(2022)

The RSD_simulator package was developed to enable straightforward simulation of a broad range of ctRSD circuits. The package allows users to define specific components and concentrations to include in a simulation and then compiles these inputs into the appropriate ODEs. The relationship between inputs, gates, and outputs are assembled in matrices in the system of ODEs. DISCLAIMER: the current package has a few known limitations, which are described below. Other unknown bugs may exist, especially with some of the additional features that go beyond the scope of the current manuscript – these features have not been rigorously tested.

General overview of using the RSD simulator.py package

1. In the script you will conduct the simulations in, import the RSD_simulator like any other Python package
    a. The example scripts have a sys.path.insert() command that adds the path to where the RSD_simulator.py package is saved
        i. Users will need to update this command appropriately after downloading the package
    b. Other required packages: numpy, scipy.integrate, and matplotlib.pyplot
2. Initiate a simulation instance as: model = RSD_simulator.RSD_sim()
3. Define concentrations of components in the simulation with model.DNA_species()
4. Execute the simulation with model.simulate(t_sim,1,k_txn)

a. This call needs both a simulation time (t_sim) and a transcription rate constant (k_txn)

5. Pull out simulation results for analysis with model.output_concentration['component name']

a. Simulation time can be obtained from model.sol.t

The uploaded scripts for the main text and supplementary material simulations provide numerous examples of using the simulator. A more detailed description of the simulation options is detailed on page 3. In those scripts, RSD_simulator is imported as RSDs for simplicity.

Examples:

Supplementary Section 4 describes the ctRSD model in detail. This GitHub includes examples from the manuscript that illustrate how to initialize and use the RSD_simulator.py package. SI_figure12.py and SI_figure21.py do not use the RSD_simulator package

- figure2D_simulations.py shows how to use the simulator for a single ctRSD reaction
- figure4C_simulations.py and figure5C_simulations show how to use the simulator for OR gates
- figure4F_simulations.py shows how to use the simulator for AND gates
- figure4H_simulations.py shows how to use the simulator with a fuel strand (catalytic amplification)

- SI_figure18.py, SI_figure19.py, SI_figure26.py, SI_figure27B.py, and SI_figure30C.py show how to change specific reaction rate constants in the simulator
- SI_figure31B.py shows how to use the simulator for transcription rate calibration reactions
- discontinuous_simulation_ex.py shows how to use the simulator for stepwise discontinuous experiments, *i.e.* experiments in which a reaction proceeds for a specific amount of time and then another component is added

Current limitations of the RSD simulator.py package:

1. As described in Supplementary Section 5.2 of the ctRSD manuscript, the current software does not explicitly track which outputs come from which gates. It pools all the same outputs together regardless of which gates they came from. Thus, simulations circuits that have gates with different input domains but with the same output domains will overcount the reverse reaction between the outputs these gates. For the rate constants determined for this study, this small deviation should not change the dynamics of the systems. However, this could begin to influence things should larger circuits be simulated or should the reverse reaction rate constants change for a specific implementation.
    a. The RSD_simulator will print a WARNING when fan-in or OR circuits are programmed: 'WARNING: Fan-in (OR) circuits overestimate reverse rates'
2. Fan-out circuits are poorly supported in the current RSD_simulator package. Fan-out circuits are defined as circuits in which multiple gates that share the same input domain produce different outputs. These can be simulated the RSD_simulator package but only when the gates that execute fan-out are the same concentration. If different concentrations are specified when setting up the simulation, the concentration of the last gate defined will be used for all the fan-out gates.
    a. The RSD_simulator will print a WARNING when fan-out circuits are programmed: 'WARNING: Fan-out circuits will use the concentration of the last gate defined'
3. Reactions between fuel strand and AND gates are currently not supported. Such reactions are physically possible, but the reactions have not been included in the model equations.
    a. The RSD_simulator will print a WARNING when a fuel stand and AND gate that could react have been programmed: 'WARNING: Fuel reactions are not included for AND gates'
4. Reaction rate constants are defined based on input domains. Thus, it is only possible to change the strand displacement rate constant between and input and all gates with the corresponding input domain.
    a. For example, one can change the rate constant for I4 reacting with a 4_1 gate or a 4_2 gate, etc. It is not possible to define a rate constant for I4 reacting with a 4_1 gate and a different rate constant for I4 reacting with a 4_2 gate

Additional notes on the RSD_simulator.py package:

The RSD_simulator package is capable of simulating circuit elements beyond those described in the current manuscript:

1. Thresholding gates from: Lulu Qian and Erik Winfree, "Scaling up digital circuit computation with DNA strand displacement cascades". *Science* (2011) can be included

2. Winner-take-all (a.k.a annihilator) gates from: Kevin Cherry and Lulu Qian, "Scaling up molecular pattern recognition with DNA-based winner-take-all neural networks". *Nature* (2018) can be included

The RSD_simulator package can also simulate discontinuous experiments in which a reaction is set up with a certain set of components, allowed to proceed for a period of time, and then new component(s) are added to alter the system. Discontinuous_simulation_ex.py provides an example of this functionality.

The RSD_simulator package can also include first order degradation of all RNA components. To implement this, the user must update the degradation rate constant (kd) – this constant is set to 0 by default.

Formal description of the RSD_simulator package

A note on units:
The simulator is set up to take concentrations in nmol/L
First order rate constants need to be in units of 1/s
Second order reaction rate constants need to be in units of 1/nM-s

A note on how the model is set up:
The model is defined based on gate input domains. In the current manuscript, there are 5 unique input domains (I1, I2, I3, I4, and I5). Thus, the examples below will be described using these 5 input domains.

These examples are provided to help the user understand how the simulator operates, the actual population of the vectors and matrices is handled by the simulator itself. The user only has to supply the gate names and concentrations as described in Section 2 below.

The concentration of species is ordered by the input domain.

EXAMPLE 1: a circuit has templates for a 1_2 gate, a 3_2 gate, I1, and I3 at concentrations 25, 20, 50, and 30 nmol/L, respectively, the template concentrations will be defined as:

```
           I1  I2  I3  I4  I5
IN_con  = [50, 0 , 30, 0 , 0 ]
RSD_con = [25, 0 , 20, 0 , 0 ] (RSD_con = gate template concentrations)
```

Note in this reaction the gate concentrations are defined based on their input domains rather than their output domains.

Initial conditions for species in a simulation are defined similarly:

```
          I1  I2  I3  I4  I5
in_ic  = [0 , 0 , 0 , 0 , 0 ] (in_ic = initial concentration of input RNAs)
rsd_ic = [0 , 0 , 0 , 0 , 0 ] (rsd_ic = initial concentration of gate RNAs)
```

If the I1 RNA was initially present at 1000 nmol/L (i.e. if that RNA had been added to a sample) then the first position of the in_ic vector would be 1000.

Reaction rate constants are defined similarly. For example, for the five inputs there are five strand displacement constants for reactions with gates with corresponding input domains:

```
ksd = [ksd1, ksd2, ksd3, ksd4, ksd5]
```

Here ksd1 is the strand displacement rate constant between I1 and any gate with a 1 input domain. Likewise, ksd4 is the strand displacement rate constant between I4 and any gate with and a 4 input domain. Thus, it is only possible to change the strand displacement rate constant between and input and all gates with the corresponding input domain. See Limitation 4 above.

Matrices are used to define which outputs are produced from which gates.

The rsd_mat matrix encodes the relationship between gate input and output domains, where each column in the matrix represents an input domain and each row represents an output domain.

rsd_mat:

	I1	I2	I3	I4	I5
O1	0	0	0	0	0
O2	1	0	1	0	0
O3	0	0	0	0	0
O4	0	0	0	0	0
O5	0	0	0	0	0

EXAMPLE 2: a circuit has templates for a 1_3 gate, a 3_2 gate, and I1 concentrations 15, 25, and 50 nM, respectively, the template concentrations will be defined as:

```
           I1  I2  I3  I4  I5
IN_con  = [50, 0 , 0 , 0 , 0 ]
RSD_con = [15, 0 , 25, 0 , 0 ] (RSD_con = gate template concentrations)
```

rsd_mat:

	I1	I2	I3	I4	I5
O1	0	0	0	0	0
O2	0	0	1	0	0
O3	1	0	0	0	0
O4	0	0	0	0	0
O5	0	0	0	0	0

EXAMPLE 3: A 3&1_2 gate template at 25 nmol/L with I1 and I3 templates at 25 nmol/L each.

The RSD_conA vector holds the AND gate concentrations. These are defined by the first input the AND gate takes. For a 3&1_2 gate, the first input domain is the 3-domain so the AND gate concentration is stored in the third position of the RSD_conA vector.

```
            I1  I2  I3  I4  I5
IN_con  =  [50, 0 , 0 , 0 , 0 ]
RSD_conA = [0, 0 , 25, 0 , 0 ] (RSD_conA = AND gate template concentrations)
```

For AND gates, two inputs are required to produce an output. Thus, two rsd matrices were used to define how inputs relate to AND gate actuation and signal release. rsd_matA1 encodes the relationship between the first input and the second input in the gate. For the 3&1_2 gate, I3 opens the gate for I1.

rsd_matA1:

	I1	I2	I3	I4	I5
O1	0	0	1	0	0
O2	0	0	0	0	0
O3	0	0	0	0	0
O4	0	0	0	0	0
O5	0	0	0	0	0

rsd_matA2 encodes the relationship between the second input of an AND gate and the gate's output. For the 3&1_2 gate, I1 reacts with the gate after I3 (second input) to release O2.

rsd_matA2:

	I1	I2	I3	I4	I5
O1	0	0	0	0	0
O2	1	0	0	0	0
O3	0	0	0	0	0
O4	0	0	0	0	0
O5	0	0	0	0	0

Finally, because the AND gates are described by their first input domain, another matrix is needed to encode the relationship between and AND gate and its leak output product. For the 3&1_2 gate, the gate is defined by its first input (I3) and produces O2 as a leak output.

rsd_matAL:

	I1	I2	I3	I4	I5
O1	0	0	0	0	0
O2	0	0	1	0	0
O3	0	0	0	0	0
O4	0	0	0	0	0
O5	0	0	0	0	0

1. Class initialization definition:

CLASS: RSD_sim(domains=5)

Initializes the model instance for simulation

model = RSD_simulator.RSD_sim( ) defines the model instance with the default number of domains. The variable "model" can be named anything but will be used as "model" throughout this documentation.

Input definitions:

For domains:

The number of unique input domains in the system

This input is optional unless more than 5 input domains are being used. Simulating circuits will fewer than 5 input domains is fine with this default setting

Output definitions:

After initializing the model, the user has access to the following Attributes. These Attributes will be updated as the DNA_species( ) function is used to define the model instance. These can be obtained *via* following commands:

model.k - the number of unique input domains in the model
model.IN_con - the concentration of input templates, ordered by input domain
model.RSD_con - the concentration of single-input gate templates, ordered by input domain
model.RSD_conA – the concentration of AND gate templates, ordered by the first input domain
model.F_con - the concentration of fuel strand templates, ordered by gate input domain
model.WTA_con - the concentration winner-take-all gate templates, ordered by the first defined input domain
model.TH_con - the concentration of threshold gate templates, ordered by input domain
model.OUT_con - the concentration of templates the produce the output strand directly, ordered by output domain

model.in_ic - initial concentration of input RNAs, ordered by input domain
model.rsd_ic - initial concentration of single-input gate RNAs, ordered by input domain
model.rsdA_ic - initial concentration of AND gate RNAs, ordered by the first input domain
model.REP_con - initial concentration of DNA reporters, ordered by output domain
model.f_ic - initial concentration of fuel RNAs, ordered by gate input domain
model.wta_ic - initial concentration of winner-take-all gate RNAs, ordered by the first defined input domain

model.th_ic - initial concentration of threshold gate RNAs, ordered by input domain
model.out_ic - initial concentration of output RNAs, ordered by output domain

model.rsd_mat - a matrix defining which single-input gates produce which outputs, columns are input domains and rows are output domains. A 1 indicates a gate that has been defined
model.rsd_matA1 - a matrix defining which input domain is exposed when an AND gate reacts with its first input, columns are input domains and rows are output domains. A 1 indicates a gate that has been defined
model.rsd_matA2 - a matrix defining which output domain is released when an AND gate reacts with its second input, columns are input domains and rows are output domains. A 1 indicates a gate that has been defined
model.rsd_matAL - a matrix defining which output domain is produced *via* leak from an AND gate, columns are the first input domains and rows are output domains. A 1 indicates a gate that has been defined
model.wta_mat - a matrix defining which output domains react with a winner-take-all gate. Columns are the first defined output and rows are the second defined output

The below vectors are used to define the matrices above but provide no further information than the matrices. These intermediate vectors are not strictly necessary.

model.rsd_vec - a vector defining which single-input gates produce which outputs, ordered by input domain
model.rsd_vecA1 - a vector defining which input domain is exposed after an AND gate reacts with its first input, ordered by input domain
model.rsd_vecA2 - a vector defining which output is released after an AND gate reacts with its second input, ordered by input domain
model.wta_vec - a vector defining which outputs react with a winner-take all gate, ordered by output domain

2. Function: DNA_species( ) definition:

```
model.DNA_species(comp,name,temp_con=0,ic=0)
--------------------------------------------------------------------------------
```

Defines a component (comp) in the model instance. Concentrations should be in nmol/L. DNA_species( ) can be called as many times as needed to specify all the circuit components for a given simulation instance

Input definitions:

For comp:

String designating the component type to add to the model instance

Options:

'input' – an RNA input strand
'fuel' – an RNA fuel strand
'reporter' – a fluorescent DNA reporter complex
'threshold' – a threshold complex from Qian and Winfree (2011)
'output' - an RNA output strand
'gate' – a ctRSD gate

This includes single-input gates, AND gates, and winner-take-all gates (Cherry and Qian, 2018)

For name:

String designating the name of the component specified in comp

For comp = 'input', name = 'I1' or 'I3', etc., for input 1 or input 3

For comp = 'fuel', name = 'F1' or 'F3', etc., for fuel strands that react with gates with 1 or 3 input domains, respectively

For comp = 'reporter', name = 'REP2' or 'REP4', etc., for reporters that react with output 2 or output 4, respectively

For comp = 'threshold', name = 'TH1' or 'TH2', etc., for threshold gates that react with input/output 1 or input/output 2, respectively

For comp = 'output', name = 'OUT2' or 'OUT3' ('O2' or 'O3'), etc., for output domains 2 or 3, respectively

Note: All output -> reporter reactions are modeled as irreversible so there is no such thing as O2r in the model (in the manuscript O2r reacts irreversibly with REP2 and O2 reacts reversibly. In the model O2 is modeled as O2r and this is the same for all outputs, they are modeled to react irreversibly with their reporters

For comp = 'gate'

For a single input gate, name = '1_2' or '3_4' or '5_1', etc., for a 1_2 gate, a 3_4 gate, and a 5_1 gate, respectively. The first number represents the input domain of the gate and the second number represents the output domain of the gate

For an AND gate, name = '3&1_2' or '5&4_1', etc. The first number represents the input domain for the first input of the gate, the second number represents the input domain for the second input of the gate. The third number represents the output domain of the gate

For a winner-take-all (WTA) gate, name = '2_4 WTA' or '1_3 WTA', etc., for WTA gates that react with outputs 2/4 or outputs 1/3, respectively

For temp_con:

A number defining the template concentration (nmol/L) for the model component specified by the comp and name variables

This is set to a default value of 0 nmol/L unless the user supplies otherwise

For ic:

A number defining the initial concentration (nmol/L) for the model component specified by the comp and name variables

This is set to a default value of 0 nmol/L unless otherwise specified

As an optional parameter, this variable does not need to be specified when the DNA_species( ) function is called

Output definitions:

The DNA_species( ) does not have any unique outputs but calling this function will update the Attributes of the model described in Section 1 above

3. Function: simulate( ) definition:

```
model.simulate(t_vec,iteration,k_txn,rate_constants=[],leak=0.03,leakA=0.06)

-------------------------------------------------------------------------------
Simulates the model instance for the time in t_vec using the transcription
rate constant supplied in k_txn

The simulate( ) function can be called multiple times in succession to
simulate discontinuous experiments
      Each new call needs to increase iteration by 1

leak and leakA are optional inputs that can be changed to alter the global
leak transcription rate for single-input gates (leak) and AND gates (leakA.
The default values are set to those used in the paper as percentages of the
transcription rate for each gate in the simulation (3% for single input gates
and 6% for AND gates).

The rate_constants can optionally be changed
      The default values are:


      ksd    = 1e-6 (L/nmol-s) Forward strand displacement rate constant
                               between gates and inputs


      kfsd   = 1e-6 (L/nmol-s) Strand displacement rate constant between fuel
                               strands and gates

      krev   = 2.7e-7 (L/nmol-s) Reverse strand displacement rate constant
                                 between gates and their outputs for all
                                 outputs but O2

      krev2  = 5e-9 (L/nmol-s) Reverse strand displacement rate constant
                               O2 and a gate that releases O2

      kf_rep = 1e-5 (L/nmol-s) Forward strand displacement rate constant
                               between an output and a reporter

      kr_rep = 0 (L/nmol-s) Reverse strand displacement rate constant
                            for a reporter

      kf_wta = 1e-3 (L/nmol-s) Forward strand displacement rate constant
                               between an output and a winner-take-all gate

      kr_wta = 0.4 (1/s) Reverse strand displacement rate constant
                         between an output and a winner-take-all gate

      kRz    = 4.167e-3 (1/s) Ribozyme cleavage rate constant

      kth    = 1e-4 (L/nmol-s) Thresholding gate rate constant

      kd     = 0 (1/s) First order RNA degradation rate constant
```

Input definitions:

For t_vec:

A numpy array specifying the time values (in seconds) to run the simulation for (*i.e.* t_vec = numpy.linspace(0,3,1000)*3600)

For iteration:

Increment each time the simulate( ) function is called in a discontinuous simulation

See the discontinuous_simulation_ex.py example for how to use this

For most simulations this will be 1

For k_txn:

The first order transcription rate to use in the simulation (in 1/s)

For rate_constants:

An optional input that allows the user to change the rate constant values in a specific simulation instance

The default rate constant values are tabulated above

Rate constant vectors are set up as below for a 5 input system:

ksd = [ksd1, ksd2, ksd3, ksd4, ksd5]

Here ksd1 is the strand displacement rate constant between I1 and any gate with a 1 input domain. Likewise, ksd4 is the strand displacement rate constant between I4 and any gate with and a 4 input domain. Thus, it is only possible to change the strand displacement rate constant between and input and all gates with the corresponding input domain. See Limitation 4 above.

When changing a single rate constant, all other rate constants must be supplied (even at the default values)

EXAMPLE: Below all the rate constants are supplied at the default values and the reverse RNA strand displacement rate constant for I2 domains has been changed. This is the default 5 input system so there are 5 possible krev rates so all five have to be specified to change one of them. Note for the other rate constants that take the default values only a single value is needed and that value will be used for all five input domains

```
ksd = 1e3/1e9
kfsd = 1e3/1e9
krev = 270/1e9
kf_rep = 1e4/1e9
kr_rep = 0*1e2/1e9
kf_wta = 1e5/1e9
kr_wta = 0.4
kRz = 0.25/60
kth = 1e5/1e9
```

```
kd = 0

k_txn = 0.015

rte=[[k_txn],[ksd],[kfsd],[krev,1e6/1e9,krev,krev,krev],[kf_rep],[kr_re
p],[kf_wta],[kr_wta],[kRz],[kth],[kd]]

model.simulate(t_vec,iteration,k_txn,rate_constants=my_rte)
```

For leak and leakA:

Optional inputs that allow the user to define the leak transcription rate as a percentage of the global transcription rate for each gate in the simulation

The default values are 3% (0.03) for single input gates and 6% (0.06) for AND gates

These inputs can be supplied as a single number, a list with a single value, or a list of values with a leak percentage for each domain in the system (default is 5 domains). So for the default system, a list entry for this input must either have one entry that will be applied to all domains or an entry for each individual domain. The same applies for the leakA input

EXAMPLES:

```
model.simulate(t_vec,iteration,k_txn,leak=0)
```

This sets the leak percentage to 0% for all domains in the simulation

```
model.simulate(t_vec,iteration,k_txn,leak=[0.05])
```

This sets the leak percentage to 5% for all domains in the simulation

```
model.simulate(t_vec,iteration,k_txn,leak=[0.05,0.03,0,0.03,0.03])
```

This sets the leak percentage to 5% for domain 1 and 0% for domain 3 and 3% for domains 2, 4, and 5 in the simulation. Note all 5 default domains must be defined to change the leak percentage for individual domains

Output definitions:
After executing the simulate( ) function, concentrations of different species in the simulation can be obtained with the following commands:

```
CON = model.output_concentration['Component Name']
```

The component names are as follows:

```
'IN'      - input RNAs
'RSDg'    - cleaved single input gates
'OUT'     - output RNAs
'REP'     - DNA reporters
'RSDgA1'  - cleaved, unreacted AND gates
'RSDgA2'  - cleaved AND gates that have reacted with their first input
'I_RSDg'  - input:single-input gate complexes
'F'       - Fuel RNAs
'F_RSDg'  - fuel:single-input gate complexes
'uRSDg'   - uncleaved single-input gates
'uRSDgA'  - uncleaved AND gates
'uWTA'    - uncleaved winner-take-all gates
'WTA'     - cleaved, unreacted winner-take-all gates
'WTA1'    - cleaved winner-take-all gates that have reacted with their first
            defined input
'WTA2'    - cleaved winner-take-all gates that have reacted with their second
            defined input
'uTH'     - uncleaved threshold gates
'TH'      - cleaved, unreacted threshold gates
'O_RSDg'  - cleaved single-input gates that have reacted with an output from
            another gate
'I_RSDgA'- input:AND gate complexes
'O_RSDgA'- cleaved AND gates that have reacted with an output from another
            gate
```

The input domain of the specific component desired needs to be supplied with the component name. The component names are defined by their input domains.

EXAMPLES:

To obtain the I1 concentrations from the simulations:

```
I1_con = model.output_concentration['IN1']
```

To obtain the O2 concentrations from the simulations:

```
I1_con = model.output_concentration['OUT2']
```

To obtain the REP2 concentrations from the simulations:

```
I1_con = model.output_concentration['REP2']
```

To obtain the concentration of uncleaved 1_4 gate from the simulations:

```
I1_con = model.output_concentration['uRSDg1']
```

To obtain the concentration of cleaved 3_2 gate from the simulations:

```
I1_con = model.output_concentration['RSDg3']
```

After executing the simulate( ) function the user also has access to the following model Attributes:

model.sol – the solution object from the ODE simulator
model.sol.t provides the time values from the simulation
model.sol.y provides the component concentrations from the simulation

The model reaction rate constants used in the simulation, ordered from input 1 to input N:

model.k_txn
model.ksd
model.kfsd
model.krev
model.kf_rep
model.kr_rep
model.kf_wta
model.kr_wta
model.kRz
model.kth
model.kd