Exercises[1]

October 6, 2021

[1] HMS, 2021, v1.1

Exercise 1: Find the bug in this model

Find the bugs in this model

```
import tellurium as te

r = te.loada ('''
      $Xo -> S1;  k1*Xo;
       S2 -> S3;  k2*S1;
       S3 -> $X1; k3*S2;

       Xo = 10; k1 = 0.4; k2 = 0.23; k3 = 0.13;
''')

m = r.simulate (0, 60, 100)
r.plot()
```

1. Run a simulation from 0 to 60
2. What do you observe? Do do think the simulation is correct?
3. If not, where is the bug(s) in the model?

Exercise 2: Basic Concepts

Consider the following model and answer the questions.

```
import tellurium as te

r = te.loada ('''
      $Xo -> S1;  k1*Xo;
       S1 -> S2;  k2*S1;
       S2 -> $X1; k3*S2;

       Xo = 10; k1 = 0.4; k2 = 0.23; k3 = 0.13;
''')
```

1. Identify the boundary species
2. Identify the floating species
3. Run a simulation from 0 to 60 time units

Exercise 3: Steady State

Setting and getting values:

```
import tellurium as te

r = te.loada ('''
      $Xo -> S1;  k1*Xo;
       S1 -> S2;  k2*S1;
       S2 -> $X1; k3*S2;

       Xo = 10; k1 = 0.4; k2 = 0.23; k3 = 0.13;
''')
```

1. Run a simulation from 0 to 60 using 100 points
2. What is happening towards the end of the simulation to S1 and S2?
3. Type `r.getRatesOfChange()` at the **console**, what do you see?
4. Type `r.getFloatingSpeciesIds()` at the **console**, the order of the names correspond to the order in the getRatesOfChange array.
5. Type `r.steadyState()` to compute the steady state directly.

Exercise 4: Set, get and reset values

Setting and getting values:

```
import tellurium as te

r = te.loada ('''
      $Xo -> S1;  k1*Xo;
       S1 -> S2;  k2*S1;
       S2 -> $X1; k3*S2;

       Xo = 10; k1 = 0.4; k2 = 0.23; k3 = 0.13;
''')
```

1. Type the following at the console: `r.Xo = r.Xo * 4`?
2. Enter `m = r.simulate (0, 60, 100)` at the **console**
3. Enter `r.plot()` at the **console**
4. What do you observe?
5. Type `r.S3` at the **console**

Exercise 4: Set, get and reset values

How to reset a simulation

```
import tellurium as te

r = te.loada ('''
      $Xo -> S1;  k1*Xo;
       S1 -> S2;  k2*S1;
       S2 -> $X1; k3*S2;

       Xo = 10; k1 = 0.4; k2 = 0.23; k3 = 0.13;
''')

m = r.simulate (0, 60, 100)
r.plot()
# This resets all the floating species to their original values
r.reset()

r.Xo = r.Xo * 5;
m = r.simulate (0, 60, 100)
r.plot()
```

Exercise 5: Events

Applying Perturbations to a Model: Using the `at` syntax.

```
import tellurium as te

r = te.loada ('''
      $Xo -> S1;  k1*Xo;
       S1 -> S2;  k2*S1;
       S2 -> $X1; k3*S2;

       Xo = 10; k1 = 0.4; k2 = 0.23; k3 = 0.13;
''')
```

1. Modify the model to also plot Xo as well as: time, S1 and S2
2. Add the statement: `at (time > 20):  Xo = 4*Xo` to the model
3. Run a simulation again from 0 to 60 time units

Exercise 5: Events

Answer:

```
import tellurium as te

r = te.loada ('''
      $Xo -> S1;  k1*Xo;
       S1 -> S2;  k2*S1;
       S2 -> $X1; k3*S2;

       at (time > 20): Xo = 4*Xo

       Xo = 10; k1 = 0.4; k2 = 0.23; k3 = 0.13;
''')
m = r.simulate (0, 40, 100, ['time', 'S1', 'S2', 'Xo'])
r.plot()
```

Statements like `at():` `etc` are called **events**

Exercise 5: Events

1. Add another event: `at (time > 30):  Xo = Xo / 4`
2. Rerun the simulation

Exercise 6: Tricks with Events

```
import tellurium as te

r = te.loada ('''

       Xo := sin ((time-10)*1.5*flag)

      $Xo -> S1;  k1*Xo;
       S1 -> S2;  k2*S1;
       S2 -> $X1; k3*S2;

       flag = 0
       at (time > 10): flag = 1

       k1 = 0.4; k2 = 0.23; k3 = 0.13;
''')
m = r.simulate (0, 40, 200, ['time', 'S1', 'S2', 'Xo'])
r.plot()
```

The syntax `Xo := etc` means that this equation is **part of the model**

1. Run this model, explain what it is doing
2. Add this line to the model: `at (time > 20):  flag = 0`

Exercise 7: Tricks with Events: Delayed Ramp

```
import tellurium as te

r = te.loada ('''

       Xo := (time-10)*k*flag

      $Xo -> S1;  k1*Xo;
       S1 -> S2;  k2*S1;
       S2 -> $X1; k3*S2;

       flag = 0
       at (time > 10): flag = 1

       k = 0.5; k1 = 0.4; k2 = 0.23; k3 = 0.13;
''')
m = r.simulate (0, 40, 200, ['time', 'S1', 'S2', 'Xo'])
r.plot()
```

1. Run this model, explain what it is doing

Exercise 8: Tricks with Events: Different Signals

```
import tellurium as te

r = te.loada ('''
     # All waves have the following amplitude and period
     amplitude = 1
     period = 10

     # These events set the 'UpDown' variable to 1 or 0 according to the period.
     UpDown=0
     at sin(2*pi*time/period) >  0, t0=false: UpDown = 1
     at sin(2*pi*time/period) <= 0, t0=false: UpDown = 0

     # Simple Sine wave with y displaced by 3
     SineWave := amplitude/2*sin(2*pi*time/period) + 3

     # Square wave with y displaced by 1.5
     SquareWave := amplitude*UpDown + 1.5

     # Triangle waveform with given period and y displaced by 1
     TriangleWave = 1
     TriangleWave' = amplitude*2*(UpDown - 0.5)/period

     # Saw tooth wave form with given period
     SawTooth = amplitude/2
     SawTooth' = amplitude/period
     at UpDown==0: SawTooth = 0

     # Simple ramp
     Ramp := 0.03*time
''')
result = r.simulate (0, 90, 500)
r.plot()
```

1. Run this model