Getting started with the E. Coli whole cell model: A brief tutorial

Basics

To run a simulation without saving the output, usually just for debugging purposes, if you already have a fit knowledge base:

```
make justSimulation
```

To only make a fit knowledge base:

```
make justKb
```

The knowledge base and the code that fits the knowledge base is located in the reconstruction/ecoli/ directory.

Processes are located in wcEcoli/models/ecoli/processes/

Each process has three components: initialize (called only once at the beginning of a simulation), calculateRequest (called at the beginning of each timestep), and evolveState (called after resources are allocated at each timestep).

In initialize: get needed parameters from the knowledge base, get views of bulk and unique molecules (bulk molecules are "indistinguishable" from each other, e.g. inactive RNAP molecules, unique molecules can be distinguished from each other, e.g. active RNAP molecules are each assigned to a specific transcript), create a view so that you can get counts, change counts, change properties

In calculateRequest: request the resources that you want for that timestep (don't request all unless you are certain that another process doesn't need this resource as well, don't forget about metabolism)

In evolveState: resources have been allocated, perform the process, must update masses (must be conserved), update counts

If you need new output, you must write this out as a listener. To add a listener, add the file to wcEcoli/models/ecoli/listeners/, add the listener to listeners and _listenerClasses in wcEcoli/models/ecoli/sim/simulation.py

To change the cell's initial conditions:
Change wcEcoli/models/ecoli/sim/initial_conditions.py

To load the knowledge base into ipython:

```
ipython
import reconstruction.ecoli.knowledge_base as kbe
kb=kbe.KnowledgeBaseEcoli()
```

Example: varying transcription rate

Background

Genes encoding for stable RNA are transcribed at a faster rate in E. Coli. In the model (prior to this tutorial), all genes were transcribed at the same rate of 42nt/s. In this tutorial, we will change the model so that genes for rRNA and tRNA are transcribed at 80nt/s while everything else is still transcribed at 80nt/s.

The code has inevitably evolved since I wrote this tutorial, so let's get on the same page by going back to an older version of the code. Let's do this tutorial on a new branch. cd into your wcEcoli directory and type:

```
git checkout –b tutorial d640563
```

Add a new parameter to the model

First, let's add a new elongation rate to the knowledge base. In reconstruction/ecoli/knowledge_base.py add to the _loadParameters function (after line 1598):

```
self._parameterData['rnaPolymeraseElongationRateFast']=80*units.nt/units.s
```

Create a new process

Let's split the transcript elongation process into a slow process and a fast process. Go to models/ecoli/processes/ :

```
cd models/ecoli/processes/

cp transcript_elongation.py transcript_elongation_fast.py

mv transcript_elongation.py transcript_elongation_slow.py
```

Now we need to add these processes to the simulation, so that they actually get called. Let's go back to the wcEcoli directory. Open models/ecoli/sim/simulation.py and change line 19 to read:

```
from models.ecoli.processes.transcript_elongation_slow import
TranscriptElongationSlow
```

Then add a line just under this:

```
from models.ecoli.processes.transcript_elongation_fast import
TranscriptElongationFast
```

Under process classes, add TranscriptElongationFast and change TranscriptElongation to TranscriptElongationSlow. _processClasses should now look like this:

```
        _processClasses = (
                Metabolism,
                RnaDegradation,
                TranscriptInitiation,
```

```
TranscriptElongationSlow,
TranscriptElongationFast,
PolypeptideInitiation,
PolypeptideElongation,
Replication,
ProteinDegradation,
Complexation,
AtpUsage
)
```

Change the model output

Now let's change the listeners to record output from these new processes. The only listener that takes output from TranscriptElongation is ntp_usage, so let's modify this. Open models/ecoli/listeners/ntp_usage.py and go to line 50. Change this line to read:

```
self.transcriptionProcessSlowIdx =
sim.processes.keys().index("TranscriptElongationSlow")
```

Under this, let's add a process index for the fast transcription:

```
self.transcriptionProcessFastIdx =
sim.processes.keys().index("TranscriptElongationFast")
```

In the update function in this file, let's change the calculation of NTPUsageCurrent. Change it to read:

```
self.transcriptionNtpUsageCurrent =
(self.bulkMolecules._countsAllocatedInitial[self.metaboliteIdxs,
self.transcriptionProcessSlowIdx] +
self.bulkMolecules._countsAllocatedInitial[self.metaboliteIdxs,
self.transcriptionProcessFastIdx] - self.bulkMolecules._countsAllocatedFinal[
self.metaboliteIdxs, self.transcriptionProcessSlowIdx] -
self.bulkMolecules._countsAllocatedFinal[self.metaboliteIdxs,
self.transcriptionProcessFastIdx])
```

Modify a process

Now let's modify the new processes. Open models/ecoli/processes/transcript_elongation_slow.py and change all references of TranscriptElongation to TranscriptElongationSlow. If you're using vi you can do this with the following command:

```
:%s/TranscriptElongation/TranscriptElongationSlow/g
```

Now, let's create a Boolean vector that is True at the indices of transcripts that should be elongated at the slow rate and False at the indices of transcripts that should be elongated at the fast rate. In the initialize function add the following line (after line 59):

```
self.slowRnaBool = ~(kb.rnaData["isRRna5S"] | kb.rnaData["isRRna16S"] |
kb.rnaData["isRRna23S"] | kb.rnaData["isTRna"])
```

“isRRna5S”, “isRRna16S”, “isRRna23S”, and “isTRna” are properties of rnaData, which is found in the knowledge base (kb). These are vectors of Boolean values. This creates a vector that is False for all 5S rRNA, 16S rRNA, 23S rRNA, and tRNA, and True for everything else. Now let’s change the view onto the active rna polymerases to a view onto only the active rna polymerases that are on transcripts that should be elongated at the slow rate. Change the definition of self.activeRnaPolys to read:

```
self.activeRnaPolys = self.uniqueMoleculesView(
      'activeRnaPoly',
      rnaIndex = ("in", np.where(self.slowRnaBool)[0])
      )
```

This is all we need to change in this file. Now let’s open the fast transcript elongation file models/ecoli/processes/transcript_elongation_fast.py. Similarly to what we did in the slow elongation file, let’s change all references of TranscriptElongation to TranscriptElongationFast. If you’re using vi you can do this with the following command:

```
:%s/TranscriptElongation/TranscriptElongationFast/g
```

Now, let’s create the opposite of the Boolean vector that we created in the transcript_elongation_slow. In the initialize function add the following line (after line 59):

```
self.fastRnaBool = kb.rnaData["isRRna5S"] | kb.rnaData["isRRna16S"] |
kb.rnaData["isRRna23S"] | kb.rnaData["isTRna"]
```

Now let’s change the view onto the active rna polymerases to a view onto only the active rna polymerases that are on transcripts that should be elongated at the fast rate:

```
self.activeRnaPolys = self.uniqueMoleculesView(
        'activeRnaPoly',
        rnaIndex = ("in", np.where(self.fastRnaBool)[0])
        )
```

Finally, let’s change the elongation rate. Change line 59 to read:

```
self.elngRate = kb.rnaPolymeraseElongationRateFast.asNumber(units.nt /
units.s) * self.timeStepSec
```

Now we’re done changing this file as well.

Changing the fitting

Now we could run a simulation and rRNA and tRNA would be transcribed at the new fast rate, while everything else was transcribed at the same slow rate before. (Try it: `make runSimulationJob WC_LOGTODISKEVERY=10 DESC="Tutorial, rRNA and tRNA is transcribed at a faster rate, but nothing has been changed in the way the simulations are initialized"` and specifically look at the rnapActiveFraction plot. This should be around 20%. Is it?) However, the simulation would be initialized the same way as before. This is a problem because before running a simulation, we need to fit several initial parameters including the initial counts of RNA polymerases and the activation rate of the RNA

polymerases. Because some things are now being transcribed at a faster rate, we should need fewer RNA polymerases to make the same number of transcripts. Additionally, if some things are being transcribed more quickly, then RNA polymerases will be inactivating more quickly. In order to maintain a constant ratio of active to inactive RNA polymerases (experimentally shown to be around 20%), the activation rate must also increase now.

We calculate the number of RNA polymerases that we need initially by writing an equation for the rate of change of some RNA R_i (rate of synthesis – rate of degradation = rate of dilution due to growth):

$$\frac{dR_i}{dt} = \frac{k_{elong}}{L_i} P_i(t) - \frac{\ln(2)}{h_i} R_i(t) = \frac{\ln(2)}{\tau_d} R_i(t)$$

L_i is the length of transcript i; P_i is the number of RNA polymerases actively transcribing R_i at time t; h_i is the half life of R_i; τ_d is the length of the cell cycle; k_{elong} is the transcript elongation rate. Right now this is a constant value for all transcripts. We want to change this to be different for rRNA and tRNA. Let's change this in the code. Open the file reconstruction/ecoli/fitkb1.py and go to the setRNAPCountsConstrainedByPhysiology function. This is what we need to change. Let's add the Boolean vectors for fast and slow transcripts. After line 300 add these two lines:

```
slowRnaBool = ~(kb.rnaData["isRRna5S"] | kb.rnaData["isRRna16S"] |
kb.rnaData["isRRna23S"] | kb.rnaData["isTRna"])

fastRnaBool = kb.rnaData["isRRna5S"] | kb.rnaData["isRRna16S"] |
kb.rnaData["isRRna23S"] | kb.rnaData["isTRna"]
```

Let's break up the calculation of nActiveRnapNeeded into nActiveRnapNeededforFast and nActiveRnapNeededforSlow. Change the nActiveRnapNeeded calculation to read:

```
nActiveRnapNeededforSlow =
calculateMinPolymerizingEnzymeByProductDistribution(rnaLengths[slowRnaBool],
kb.rnaPolymeraseElongationRate, rnaLossRate[slowRnaBool],
rnaCounts[slowRnaBool])
```

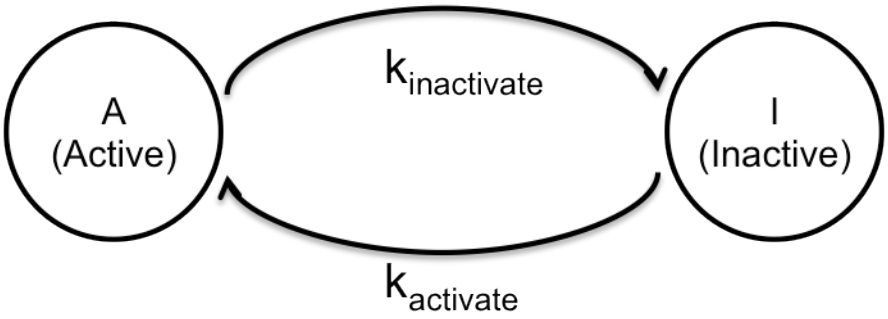
Add a line under this for the calculation for the fast transcripts:

```
nActiveRnapNeededforFast =
calculateMinPolymerizingEnzymeByProductDistribution(rnaLengths[fastRnaBool],
kb.rnaPolymeraseElongationRateFast, rnaLossRate[fastRnaBool],
rnaCounts[fastRnaBool])
```

Below this, add a line to calculate the total number of RNA polymerases needed:

```
nActiveRnapNeeded = nActiveRnapNeededforFast + nActiveRnapNeededforSlow
```

Now let's change the calculation of the RNA polymerase activation rate. In the model, an RNA polymerase can either be active or inactive:



To calculate the rate of activation, $k_{activate}$, we write an equation for the rate of change of the number of RNAP active at steady state:

$$\frac{dA}{dt} = k_{activate} I - k_{inactivate} A = 0$$

We know the experimentally determined fraction of RNA polymerases that are active, f_{active}:

$$f_{active} = \frac{A}{A + I}$$

Rearranging and solving for the rate of activation:

$$k_{activate} = k_{inactivate} \left(\frac{f_{active}}{1 - f_{active}} \right)$$

We can calculate the rate of inactivation from the elongation rate, the transcript lengths, and the synthesis probabilities:

$$k_{inactivate} = \frac{k_{elong}}{\boldsymbol{L_i}} \cdot \frac{1}{\boldsymbol{p_{synth}}}$$

We want to change the elongation rate to be a vector, rather than a constant:

$$k_{inactivate} = \frac{\boldsymbol{k_{elong,i}}}{\boldsymbol{L_i}} \cdot \frac{1}{\boldsymbol{p_{synth}}}$$

Now let's do this in the code. In reconstruction/ecoli/fitkb1.py, in the fitRNAPolyTransitionRates function, at the top add these lines:

```
slowRnaBool = ~(kb.rnaData["isRRna5S"] | kb.rnaData["isRRna16S"] |
kb.rnaData["isRRna23S"] | kb.rnaData["isTRna"])

fastRnaBool = kb.rnaData["isRRna5S"] | kb.rnaData["isRRna16S"] |
kb.rnaData["isRRna23S"] | kb.rnaData["isTRna"]
```

Now, delete the elngRate definition (elngRate = kb.rnaPolymeraseElongationRate) and add an elongation rate vector:

```
elngRateVector = slowRnaBool*kb.rnaPolymeraseElongationRate +
fastRnaBool*kb.rnaPolymeraseElongationRateFast
```

Delete the calculation of averageTranscriptLength and expectedTerminationRate and add the following three lines:

```
expectedTranscriptionTime = rnaLengths/elngRateVector

weightedExpectedTranscriptionTime = units.dot(expectedTranscriptionTime,
synthProb)

expectedTerminationRate = 1/weightedExpectedTranscriptionTime
```

We're done editing this file now. Now the fitting of the initial number of RNA polymerases and the activation rate is fixed.

Running simulations

Now we can run some simulations. If we just want to double check that things will run, we can make a fit knowledge base and start running a simulation without saving any output. Let's do this as follows:

```
make justKb
```

This will take a few minutes and should complete without errors. If you ever want to debug the knowledge base and the fitting process, this is the command that you'd want to use first. If you wanted to just debug the fitkb1 or fitkb2 process, you can run `make fitKb_1` or `make fitKb_2`, respectively.

Now let's check that a simulation will start running without errors:

```
make justSimulation
```

After about a minute, this should start producing output looking something like this:

```
covertlab-cluster (wcEcoli): make justSimulation
python2.7 setup.py build_ext --inplace
running build_ext
running build_ext
rm -fr build
PYTHONPATH="/home/users/kappel/wcEcoli:" python2.7 runscripts/execModel.py 2
"fixtures/kb" "fixtures/sim"
/opt/python2.7.2/lib/python2.7/site-packages/matplotlib/__init__.py:908:
UserWarning:  This call to matplotlib.use() has no effect
because the the backend has already been chosen;
matplotlib.use() must be called *before* pylab, matplotlib.pyplot,
or matplotlib.backends is imported for the first time.

  if warn: warnings.warn(_use_error_msg)
```

Time (s)	Dry mass (fg)	Dry mass fold change	Protein fold change	RNA fold change	Expected fold change
0	245.39	1.000	1.000	1.000	1.000
1	245.67	1.001	1.000	1.000	1.000
2	245.69	1.001	1.000	1.000	1.000
3	245.74	1.001	1.001	1.000	1.001
4	245.79	1.002	1.001	1.000	1.001
5	245.87	1.002	1.001	1.000	1.001
6	245.94	1.002	1.001	1.000	1.001
7	246.01	1.003	1.001	1.000	1.001
8	246.08	1.003	1.002	1.000	1.002
9	246.15	1.003	1.002	1.000	1.002

Now, we know that we don't have any obvious errors that prevent the simulation from running. We can just stop this. Now let's run some simulations on the cluster:

```
make runSimulationJob WC_LOGTODISKEVERY=10 DESC="Tutorial, completed"
```

This will take around 40 minutes to run. At the end you can find the output in the out/timestamp directory (timestamp obviously replaced with the actual timestamp of when you started the simulation). Look at the plots in the out/timestamp/000000/plotOut/ directory.