Comparison between ribosome flow model (RFM), TASEP (totally asymmetric exclusion process) and a Gillespie-like simulation that was developed to model ribosome dynamics on the mRNA

Peter Sarvari

19/02/2017

The steady-state equations for the ribosome-aware RFM for one transcript is:

U is the steady-state number of free ribosomes and denotes how likely it is that the ith ribosome site is occupied on the mRNA. is the initiation rate and is the averaged elongation rate on the ith ribosome site. N is the length of the open reading frame of the mRNA in terms of ribosome sites.

Note that since the transition probability depends on the probability that the next state is occupied, we cannot analyse the above model as a Markov chain, since the Markov property is not true.

Let’s reduce this n-1 equations to one nonlinear equation. This helps fsolve run faster on MATLAB and makes it less prone to numerical errors. For simplicity, first assume that . Then using the equations above, we realize that

And so on. We can easily express these terms using two recursive series, where the variable ‘a’ stands for nominator and the variable ‘b’ stands for denominator the following way:

Hence, we can write the following equation for

Solving this equation in Matlab using fsolve does not yield a meaningful solution. If the search starts from 0.2502 (explained below why) then is evaluated to be 0.2540 yielding many invalid like the ones with values bigger than one or smaller than zero. If the search starts from 0.1, then is evaluated to be 0.3189. The problem with this is that this series is extremely senitive to numerical error. This is because the with these parameters is strictly smaller than (proved below), if we require meaningful (between 0 and 1) the only possible solution when is that is very close to 1 to make sure that u is big (which it has to be since we have many total ribosomes compared to the ribosome places on the mRNA). Since , if is very close to 1, the denominator will be very close to zero, giving rise to numerical problems. Even if we assume that Matlab can handle the numbers with sufficient precision not to cause numerical issue, fsolve still cannot be used to solve the steady-state RFM, because, as we have seen above, multiple possible solutions exist, and most of them will yield invalid results. In addition, using fsolve a contradiction has arisen: using the system of n+1 nonlinear equations and solving them using fsolve actually yielded a solution that seems to be right: was chosen to be 0.2502. However, using this value (copying the value of the variable that was returned by fsolve) and substituting this back into the sole nonlinear equation that we derived that describes the system yielded that 10000 = -117.1 demonstrating how serious the numerical error is.

Proof that for , <

For < , we require the following:

Which is obviously true if is nonzero, which is the case: [1] proved persistence of the nonlinear system, which implies that any trajectory is separated from the boundary (defined as (0,1) for the probabilities) after an arbitrary short time (so obviously true for steady-state).

We also know that to make sure that is within the boundary (0,1), we require

Which yields that < 0.5.

Now if > , we have

Since (m-i) is just a dummy variable, we can rename it back to i.

This yields:

Renaming to

This is not true when

We know that c must be in (0,1). Let’s examine the minimum value of !

This function is monotone in . As , , as can be seen trivially. It is a bit more complex to find the limit as . Note that making use of the approximation

We can write

As a result

In fact, this limit is approached from above as guaranteed by the monotonicity of the function. Hence > for any valid value of . Using mathematical induction, the claim is proved.

Proposal that when , must be bigger than 0.25 if we want the steady-state u (free ribosomes) to be bigger than 0.5

For this, we combine analytical approach with numerical simulation that is used to determine the limit of the denominator series (b(i))

Using numerical simulations, we realized that only converges for (even for small i, e.g. 20 the value will be very close to the convergence value). Numerical simulations (see below) making use of the above formulas show that the limit of exists and increases until 0.25. After that diverges. The maximum limit that can be achieved is 0.5. This proves our claim.

A screenshot of a cell phone

Description generated with very high confidence

Simulation of the system using ode23 using a normal PC is close to impossible, since with the above parameters (rates are one, only one mRNA and 100 ribosome places on it), the simulation up to 500 time units (still does not correspond to steady-state) yielded an array of 2\*106 \* 101 entries, whose memory requirement is close to that of the RAM of an average PC. Furthermore, simulating the system is extremely impractical if we are only interested in the steady-state values. In addition, simulating the system for some parameter values (e.g. = 100) takes tremendous time even in this simple scenario when we only have one mRNA.

For fullness: reducing the system to one nonlinear equation in the general case

By going through the n-1 nonlinear equations from the end, making them into a form with common denominator and substituting the result into the next equation, we can derive the following series formula for the general case (when we do *not* require that )

Comparison of our developed simulation method to TASEP

The TASEP model for translation elongation as described in [2]. is very slow. One reason is that during many iterations the states do not change. Many times, elongation at a particular site is attempted, but it does not go through because there is a ribosome in the following site. This causes a huge waste of time and computational resources in the TASEP model and our developed simulation model solves this problem by not considering elongation ‘attempts’, only actual elongation steps. We believe that there is no point considering an ‘event’ that cannot happen and hence for ribosome sites, which are currently occupied by ribosomes, we define the transition probability to that site to be zero. The structure of the simulation is the usual Gillespie algorithm. This is because, once we update the transition probabilities to reflect the possible transitions and zero the probabilities for impossible transitions, we effectively obtain a continuous-time discrete-state Markov chain.

Comparison of our developed simulation method to the simulated solution of the RFM

The RFM has certain advantages: it is the mean-field approximation of the TASEP model, so we can easily calculate with the averages and there is no need to generate and ensemble or trajectories and average them to be able to observe the mean behaviour (or to simulate one trajectory long enough). However, because of the this, we are unable to characterize variance and view individual stochastic trajectories and hence observe properties that are caused by stochasticity. One more problem with the RFM, as was outlined in [2], that we neglect in the mean-field approximation that there could be correlations between sites. The independence of site is generally not true, but it is a common approximation used in the literature. Since we demonstrated earlier that fsolve cannot be used with the RFM to solve for the steady-state occupancy probabilities, we rely on ode23 to solve for the whole trajectory of the RFM. As mentioned before this is very slow. Hence, it is interesting to see whether our developed simulation method (without the mean-field approximation) executes faster than the RFM simulation for a particular example.

*Example case:*

Number of mRNA = 1  
mRNA length = 20  
total number of ribosomes = 10000  
initiation rate constant = 1  
normal elongation rate = 1  
slow elongation rate = 0.2  
slow codon location = 10th site

A screenshot of a cell phone

Description generated with high confidenceThe RFM model used here is the one described above and derived in [3] and [1]. The model described in these articles is identical for the case when we have one mRNA and the unbinding rate from the RBS is zero. The result of the RFM and our developed simulation are below. They seem to be identical (see graph) and correspond to the expected result that before the slow codon the occupation probabilities are higher than after the slow codon (not shown).

A screenshot of a cell phone

Description generated with high confidence

Our modified simulation (for 3 trajectories!) executed in 2 and a half second. The simulation for the RFM using ode23 in MATLAB took 7.85 seconds. Hence, as a conclusion based on this example (more to follow) we recommend using our developed simulation method for resource-aware translation simulation over TASEP or any version of RFM.

Comparison of the stationary distribution of our developed method to that of the analytical solution of the RFM for test cases of small mRNAs and huge initiation rates

For all the test cases:

The test cases: mRNA length of 2,3,4,7 and 19

1. One mRNA of length 2, total ribosomes = 10

From analytical solution of the steady-state RFM:

From our developed solution method, the estimated steady-state variables:

1. One mRNA of length 3, total ribosomes = 10

From analytical solution of the steady-state RFM:

From our developed solution method, the estimated steady-state variables:

1. One mRNA of length 4, total ribosomes = 10

From analytical solution of the steady-state RFM:

From our developed solution method, the estimated steady-state variables:

1. One mRNA of length 7, total ribosomes = 10

From analytical solution of the steady-state RFM:

From our developed solution method, the estimated steady-state variables:

1. One mRNA of length 19, total ribosomes = 25; first column RFM, second our method

The first column took multiple minutes to compute, whereas the second took 1-2 seconds. We see that as the length of the mRNA increases, the samples stationary distribution using our method approximates the steady-state solution of the RFM better and better.

To obtain sampling of the stationary distribution of the simulation we developed, we assumed that stationary distribution is achieved after 100 steps. We simulated for a total of 10000 steps and counted the proportions of ribosomes in a specific state between 100 and 10000 weighted by the time they were there. This is a standard technique to get the average time spent in a state in a stationary distribution. Then we divided by the total time elapsed between steps 100 and 10000 to get that on average how many ribosomes spend time in that particular state in the stationary distribution. Accordingly, if we sum these quantities for all states, we recover the total number of ribosomes.

To solve the steady-state RFM, we use MATLAB’s built-in solve function. It can only give explicit result for the mRNA of length 2 and for the others, the solution is approximated numerically at high precision using the built-in vpa function. To make the solve function run faster, we used our recursive series method (discussed earlier) to compress the several RFM equations into one.

Code: For the code, see <https://github.com/sarvarip/TranslationModels>

Acknowledgement: the developed simulation method was recommended by Matyas Juhasz, Maters student at UCL in Mathematics with Economics.

References:

[1] A. Raveh, M. Margaliot, E. D. Sontag, and T. Tuller, “A Model for Competition for Ribosomes in the Cell,” *J. R. Soc. Interface*, vol. 13, no. 116, p. 1508.02408, 2015.

[2] S. Reuveni, I. Meilijson, M. Kupiec, E. Ruppin, and T. Tuller, “Genome-scale analysis of translation elongation with a ribosome flow model,” *PLoS Comput. Biol.*, vol. 7, no. 9, 2011.

[3] R. J. R. Algar, T. Ellis, and G. B. Stan, “Modelling essential interactions between synthetic genes and their chassis cell,” *Proc. IEEE Conf. Decis. Control*, vol. 2015–Febru, no. February, pp. 5437–5444, 2014.