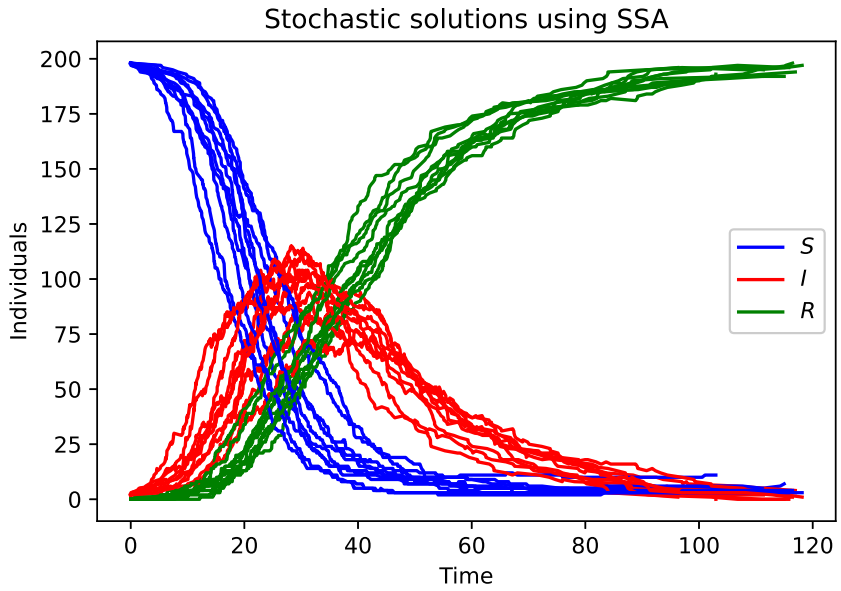
UPPSALA
UNIVERSITET

Department of Information Technology

Scientific Computing for Data Analysis

Davoud Mirzaei

Lecture 4: Stochastic Simulation Algorithm



Agenda

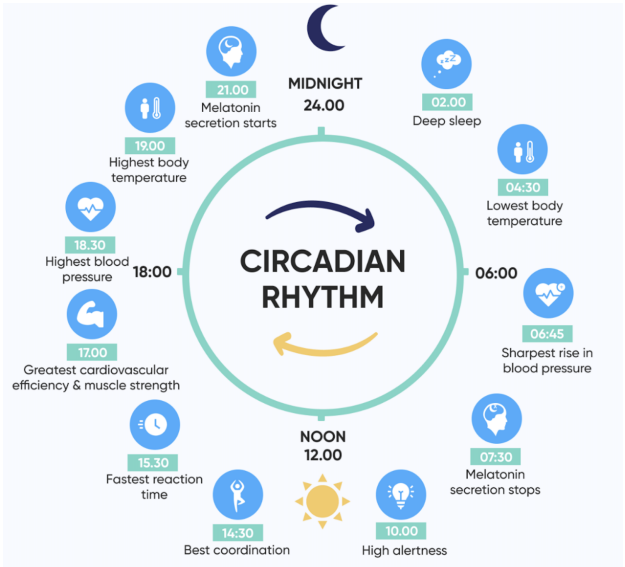
- Gillespie's algorithm (SSA)
- Application to SIR models
- Application to biochemical kinetics
- Mini-project 1

SSA

The SSA (Stochastic Simulation Algorithm) or the Gillespie algorithm[1]

- Is a stochastic method
- Is a Monte Carlo Markov process simulation
- Is used a lot in systems biology

- One example is simulation of a circadian rhythm (activator and suppressor proteins)
- Another example is simulation in epidemiology

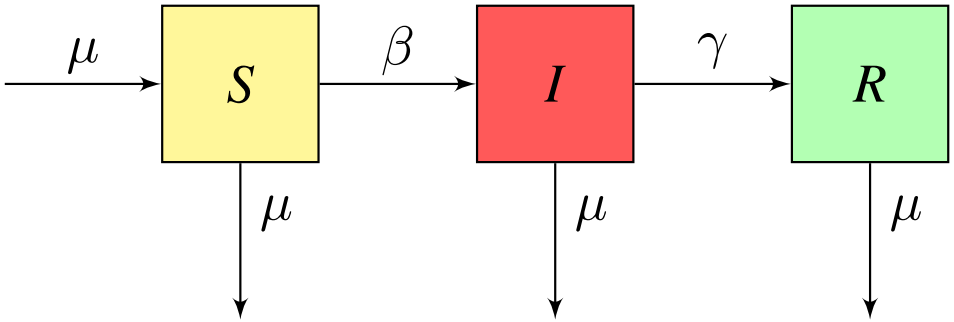


[1] Daniel T. Gillespie, *Exact Stochastic Simulation of Coupled Chemical Reactions*, The Journal of Physical Chemistry, Vol. 81, No. 25, 1977.

Image from `www.betterup.com`

Simulation of a simple epidemic model

We consider a susceptible-infected-recovered (SIR) model as below:



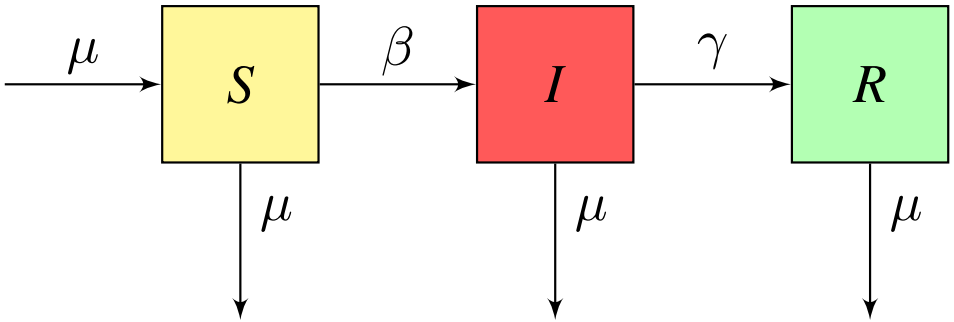
- S susceptible individuals
- I infected individuals
- R recovered individuals

unit: individual

- μ birth and death rate
- β infection rate
- γ rate of recovery

unit: $\frac{1}{\text{day}}$

Assumptions for SIR model



Some assumptions:

- $S(t)$, $I(t)$, and $R(t)$ are continuous variables of continuous time t and simulation is valid in some interval $[0, t_{final}]$
- the rates of birth and death are the same
- all newborns are susceptible
- we ignore pathogen-induced mortality
- recovered individuals are not susceptible in period $[0, t_{final}]$
- the rate of infection depends on the ratio of infected individuals to the entire population: $\beta \frac{I}{N}$

An ODE model for SIR problem

Let the total population at time t be $N(t) = S(t) + I(t) + R(t)$.

$$\frac{dS}{dt} = \mu N - \mu S - \beta \frac{I}{N} S$$
$$\frac{dI}{dt} = \beta \frac{I}{N} S - \mu I - \gamma I$$
$$\frac{dR}{dt} = \gamma I - \mu R$$

The initial condition: $S(0) = S_0$, $I(0) = I_0$, $R(0) = R_0$.

$dS(t)$ means the change in group S in interval $[t, t + dt]$ (individuals moving in minus individuals moving out in interval $[t, t + dt]$). $\frac{dS(t)}{dt}$ is the rate of change.

Deterministic solution using RK45 method

```
from scipy.integrate import solve_ivp
import numpy as np
import matplotlib.pyplot as plt

def SIR(t,y,mu,bet,gam):
    yprime = np.zeros(3);
    N = np.sum(y)
    yprime[0] = mu*N - bet*y[0]*y[1]/N-mu*y[0]
    yprime[1] = bet*y[0]*y[1]/N -(mu+gam)*y[1]
    yprime[2] = gam*y[1]-mu*y[2]
    return yprime

mu, bet, gam = 1e-4, 0.4, 0.05   # rates
tf, y0 = 120, [198,2,0]          # final time and initial values
sol = solve_ivp(SIR, [0,tf], y0, method = 'RK45', args = (mu,bet,gam))

plt.figure(figsize = (6, 3))
plt.plot(sol.t,sol.y[0],linestyle = 'solid', color='blue', label = '$S$')
plt.plot(sol.t,sol.y[1],linestyle = 'solid', color='red', label = '$I$')
plt.plot(sol.t,sol.y[2],linestyle = 'solid', color='green', label = '$R$')
plt.xlabel('time $t$'); plt.ylabel('Individuals')
plt.legend(loc = 'center right')
```

Deterministic solution using RK45 method

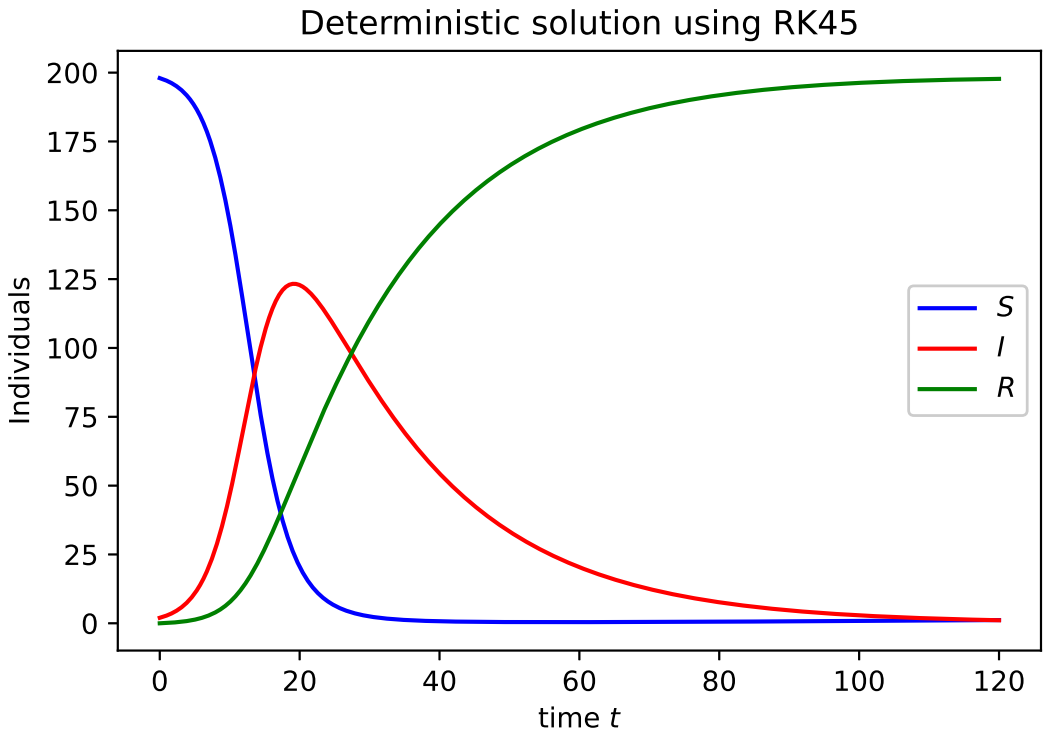
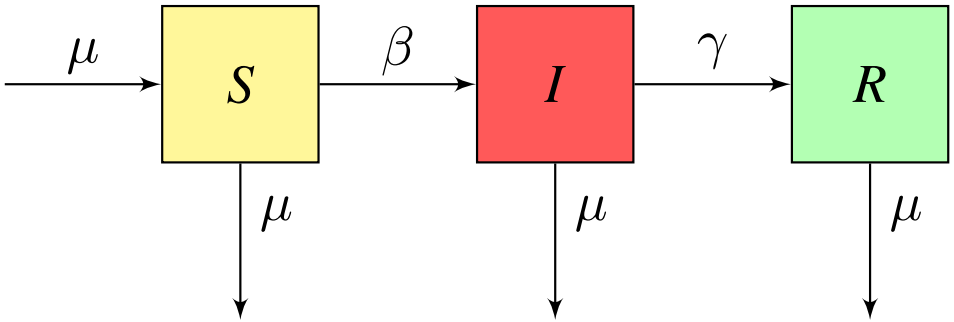


Figure: Solution of SIR model using `RK45`. The number of infected individuals increased from two initially to around 125. The epidemic reached its peak on the 20th day, and ...

Stochastic simulation of the SIR model

- Deterministic models do not account for the discrete nature of populations
- Many processes have an element of uncertainty to them. Getting infected with the virus is not the same for everyone
- To incorporate randomness into the model we should formulate the terms as probabilities at which an event occurs, and not, as in deterministic models, as rates
- In the stochastic version of the SIR model, the continuous variables $S(t)$, $I(t)$ and $R(t)$ are replaced by discrete numbers, and the rates are replaced by propensity functions (process probabilities).

SSA (or Gillespie algorithm) for the SIR model



Define the following processes (or reactions) with propensity functions (last columns)

$$
\begin{array}{lccll}
1. & \emptyset & \longrightarrow & S & w_1 = \mu N \\
2. & S & \longrightarrow & I & w_2 = \beta \frac{I}{N} S \\
3. & I & \longrightarrow & R & w_3 = \gamma I \\
4. & S & \longrightarrow & \emptyset & w_4 = \mu S \\
5. & I & \longrightarrow & \emptyset & w_5 = \mu I \\
6. & R & \longrightarrow & \emptyset & w_6 = \mu R
\end{array}
$$

Propensities can be interpreted as the *probability per unit time* that the reaction occurs

SSA or Gillespie algorithm for SIR model

Assume that we have state vector $\boldsymbol{y}(t) = (y_1(t), \ldots, y_n(t))$ and m reactions $1, 2, \ldots, m$ with propensity functions $w_1(\boldsymbol{y}), \ldots, w_m(\boldsymbol{y})$. For example, in the SIR model

$$n = 3, \quad \boldsymbol{y} = (S, I, R)$$

$$m = 6, \quad w_1 = \mu(S + I + R), \quad w_2 = \beta \frac{I}{N} S, \quad w_3 = \gamma I$$

$$w_4 = \mu S, \quad w_5 = \mu I, \quad w_6 = \mu R$$

Also assume that

$$a(\boldsymbol{y}) = \sum_{j=1}^{m} w_j(\boldsymbol{y}), \quad p_j := \frac{w_j}{a} \text{ (probabilities)}$$

Now we have the following discrete distribution table

reaction j	1	2	$\ldots$	m
probability p_j	p_1	p_2	$\ldots$	p_m

SSA or Gillespie algorithm for SIR model

- We assume that from time t to time $t+\tau$, only one reaction occurs, and this reaction results in a state change of either 0, 1, or -1.
- So, at any time t, the SSA consists of three steps:
  (1) WHEN the first reaction will occur (what is the value of τ?)
  (2) WHICH reaction will occur ($1, 2, \ldots,$ or m)
  (3) UPDATING the state
- As τ represents the waiting time for the first reaction to occur, its distribution is exponential:

$$when \sim \mathcal{E}xp\left(a(\boldsymbol{y})\right),$$

- Which reaction? Generate from the discrete distribution given above:

$$which \sim \mathcal{DD}\big([1,2,\ldots,m],[p_1,\ldots,p_m]\big)$$

(Reactions with a higher 'propensity' are more likely to happen)

SSA or Gillespie algorithm for SIR model

If *when* and *which* are sampled then the only remaining issue is how to update the state variables. We introduce the state-change vectors $\boldsymbol{v}_k$ (or stoichiometry vectors) of size n for any reaction:

$$
\begin{array}{llcll}
1. & \emptyset & \longrightarrow & S & \boldsymbol{v}_1 = [\,1,\;\; 0,\;\; 0] \\
2. & S & \longrightarrow & I & \boldsymbol{v}_2 = [-1,\;\; 1,\;\; 0] \\
3. & I & \longrightarrow & R & \boldsymbol{v}_3 = [0,\;\; -1,\;\; 1] \\
4. & S & \longrightarrow & \emptyset & \boldsymbol{v}_4 = [-1,\;\; 0,\;\; 0] \\
5. & I & \longrightarrow & \emptyset & \boldsymbol{v}_5 = [\,0,\;\; -1,\;\; 0] \\
6. & R & \longrightarrow & \emptyset & \boldsymbol{v}_6 = [\,0,\;\; 0,\;\; -1]
\end{array}
$$

In the WHICH phase if reaction k is sampled then the state is updated by

$$\boldsymbol{y}(t+\tau) = \boldsymbol{y}(t) + \boldsymbol{v}_k$$

SSA or Gillespie algorithm

Algorithm 1: Gillespie algorithm

Data: Initial state $\boldsymbol{y} = \boldsymbol{y}_0$, final time t_{final}, propensity functions $w_1, \ldots, w_m$ and state change vectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m$

Set $t = 0$;

while $t < t_{final}$ **do**

- Compute $a(\boldsymbol{y}) = w_1(\boldsymbol{y}) + \cdots + w_m(\boldsymbol{y})$ and $p_j(\boldsymbol{y}) = w_j(\boldsymbol{y})/\boldsymbol{a}_j(\boldsymbol{y})$;
- Generate $\tau \sim \mathcal{E}xp(a(\boldsymbol{y}))$;
- Generate $k \sim \mathcal{DD}\big([1, \ldots, m], [p_1, \ldots, p_m]\big)$;
- Update $t \leftarrow t + \tau$;
- Update $\boldsymbol{y} \leftarrow \boldsymbol{y} + \boldsymbol{v}_k$

The Python code is available for the SIR model from the studium:
(Modules —> Lecture Notes –> Lecture 4)
And from part 4 of Lab 1 for a Lokta-Volterra model

You can also use the code to complete mini-project 1.

The main function next page ...

SSA function with Python

```
def SSA(Initial, StateChangeMat, FinalTime):
    # Inputs:
    #  Initial: initial conditins of size (StateNo x 1)
    #  StateChangeMat: State-change matrix of size (ReactNo, StateNo)
    #  FinalTime: the maximum time we want the process be run

    # Output:
    #  AllTimes: the cell of all selected time levels
    #  AllStates: the cell of all state values at corresponding time levels

     [m,n] = StateChangeMat.shape
     ReactNum = np.array(range(m))
     AllTimes = {}   # define a list for storing all time levels
     AllStates = {}  # define a list for storing all states at all time levels
     AllStates[0] = Initial
     AllTimes[0] = [0]
     k = 0; t = 0; State = Initial
     while True:
         w = PropensityFunc(State, m)     # propensities
         a = np.sum(w)
         tau = RandExp(a,1)               # WHEN the next reaction happens
         t = t + tau                      # update time
         if t > FinalTime:
             break
         which = RandDisct(ReactNum,w/a,1)             # WHICH reaction occurs
         State = State + StateChangeMat[which.item(),] # Uppdate the state
         k += 1
         AllTimes[k] = t
         AllStates[k] = State
     return AllTimes, AllStates
```

Deterministic and stochastic solutions of SIR model

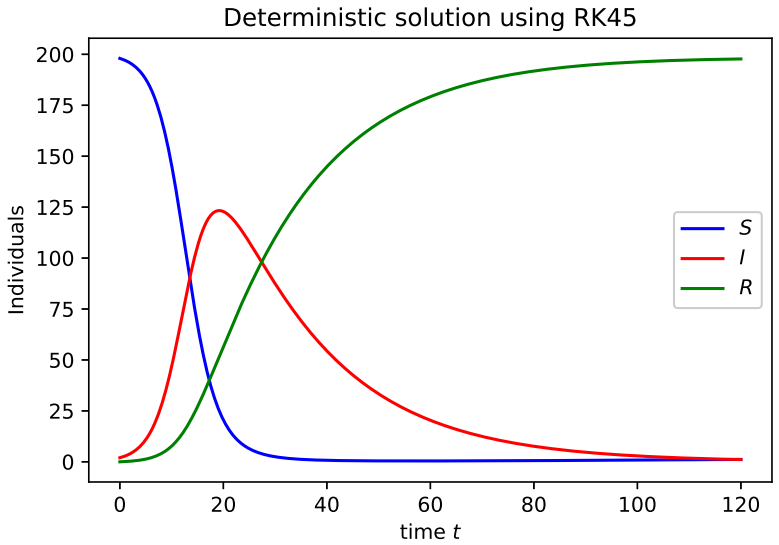
The results of 10 executions are plotted for the stochastic case. Usually, the mean of stochastic solutions is very close to the deterministic solution.

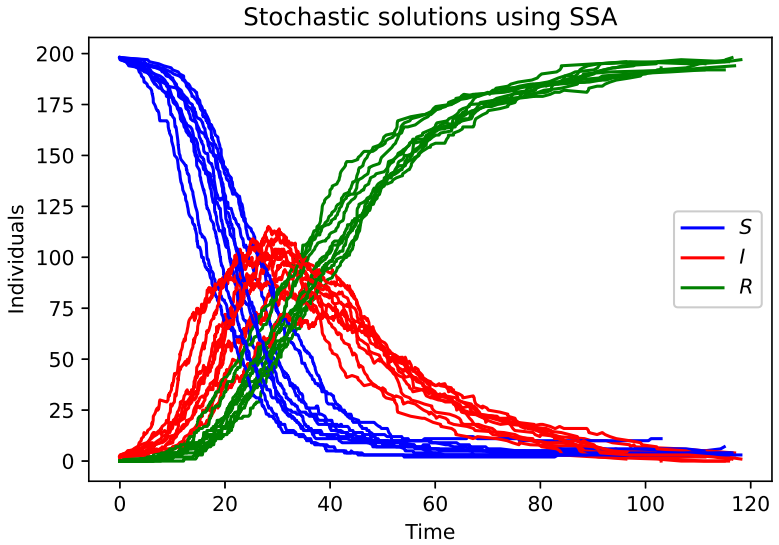
Some related questions:

What is the expected number of infected individuals on day 30?

When is the expected end of the epidemic?

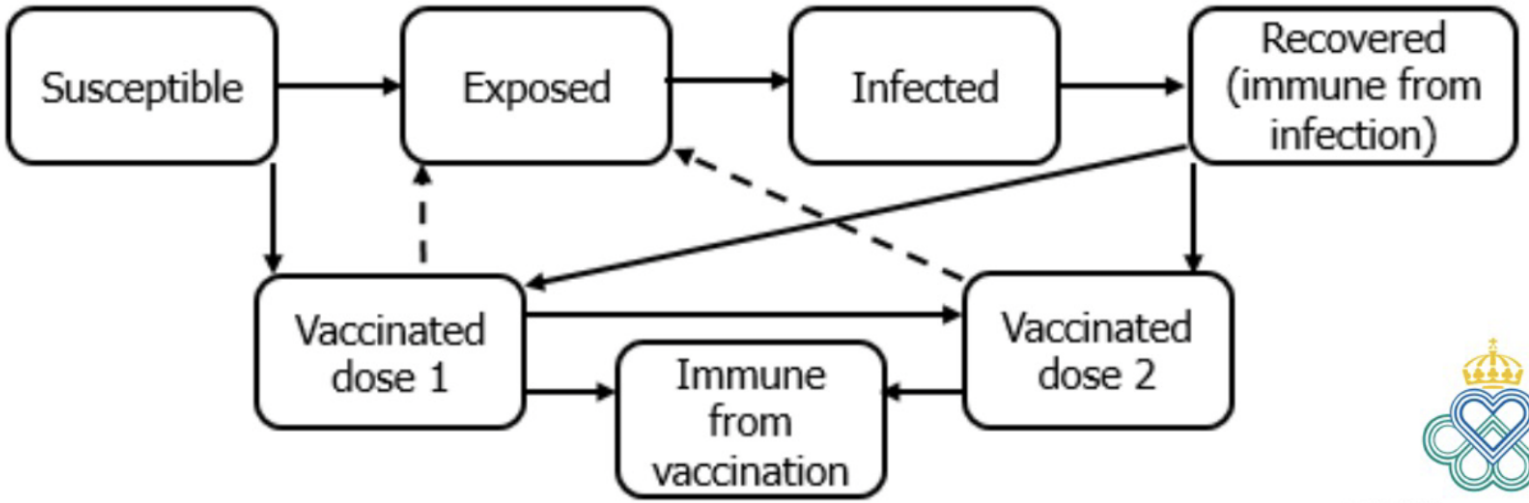
Use SSA and Monte Carlo to answer such questions.





A model for covid-19 in Sweden

FHM's scenarios for Covid-19:



Folkhälsomyndigheten

Development of the SIR model to vaccinated and immune conditions

Application to biochemical kinetics

Situation

- Situation in chemical systems formed by living cells
- finitely number of particles (instead of continuous and deterministic behaviour)
- models are usually discrete and stochastic

Model:

- n number of species $\{S_1, S_2, \ldots, S_n\}$ interacting through m chemical reactions $\{R_1, \ldots, R_m\}$
- system confined to a constant volume and

$$y_k(t) = number\ of\ molecules\ of\ species\ S_k\ at\ time\ t$$

- Goal: study evolution of state vector $\boldsymbol{y}(t) = (y_1(t), \ldots, y_n(t))$ given $\boldsymbol{y}(0) = \boldsymbol{y}_0$.

Application to biochemical kinetics

Reactions, propensity functions, state-change vectors:

- each reaction R_ℓ is "elemental" and is either unimolecular or bimolecular. Different cases: (for simplicity assume $\boldsymbol{y} = (y_1, y_2, y_3)$, here c is a constant rate)

	reaction	state-change vec.	propensity func.
case 1:	$y_1 \xrightarrow{c} y_2$	$\boldsymbol{v} = [-1, 1, 0]$	$w = cy_1$
case 2:	$y_1 \xrightarrow{c} y_2 + y_3$	$\boldsymbol{v} = [-1, 1, 1]$	$w = cy_1$
case 3:	$y_1 \xrightarrow{c} y_1 + y_2$	$\boldsymbol{v} = [0, 1, 0]$	$w = cy_1$
case 4:	$y_1 \xrightarrow{c} 2y_1$	$\boldsymbol{v} = [1, 0, 0]$	$w = cy_1$
case 5:	$y_1 \xrightarrow{c} \emptyset$	$\boldsymbol{v} = [-1, 0, 0]$	$w = cy_1$
case 6:	$y_1 + y_2 \xrightarrow{c} y_3$	$\boldsymbol{v} = [-1, -1, 1]$	$w = cy_1y_2$
case 7:	$2y_1 \xrightarrow{c} y_2$	$\boldsymbol{v} = [-1, 1, 0]$	$w = cy_1(y_1 - 1)/2$
case 8:	$2y_1 \xrightarrow{c} y_1$	$\boldsymbol{v} = [-1, 0, 0]$	$w = cy_1(y_1 - 1)/2$
case 9:	$\emptyset \xrightarrow{?} y_1$	$\boldsymbol{v} = [1, 0, 0]$	$w = ?$

- form of propensity functions follows from molecular physics and kinetic theory

Example: Chemical reactions

Michaelis-Menten system (standard model for enzyme catalysts)

$$S + E \underset{c_2}{\overset{c_1}{\rightleftharpoons}} C \xrightarrow{c_3} P + E$$

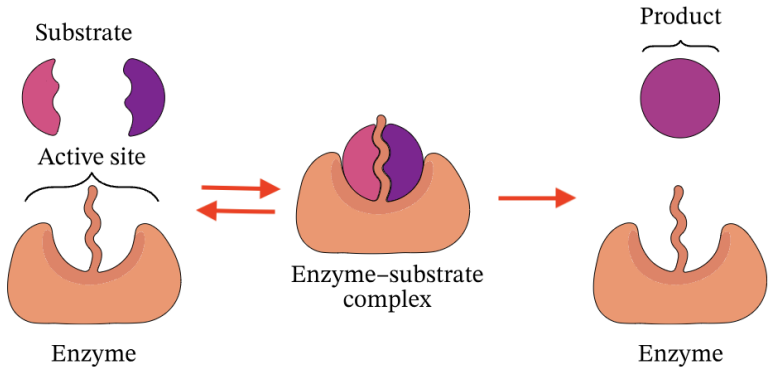
S : substrate molecules

E : enzyme molecules

C : complex of $E\&S$

P : product of the reaction

$$\boldsymbol{y} = (S, E, C, P)$$



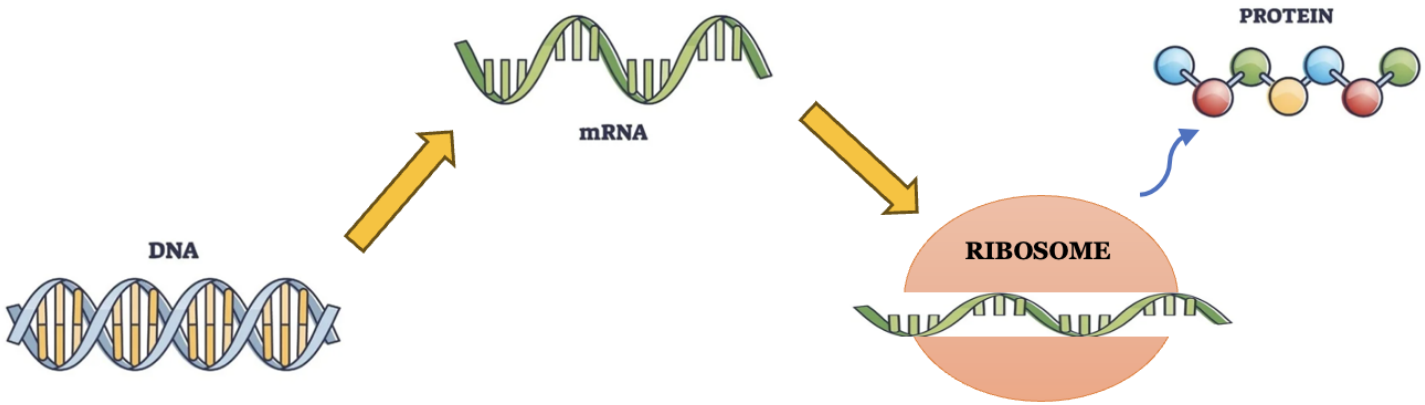
$$\begin{array}{lllll}
R_1: & S + E \xrightarrow{c_1} C & \boldsymbol{v}_1 = [-1,-1,+1,0] & w_1 = c_1 SE = c_1 y_1 y_2 \\
R_2: & S + E \xleftarrow{c_2} C & \boldsymbol{v}_2 = [+1,+1,-1,0] & w_2 = c_2 C = c_2 y_3 \\
R_3: & C \xrightarrow{c_3} P + E & \boldsymbol{v}_3 = [0,+1,-1,+1] & w_3 = c_3 C = c_3 y_3
\end{array}$$

Another example: Lotka-Volterra models. See Lab Exercise 1 for solving a Lotka-Volterra model using SSA.

image from `www.nagwa.com/en/`

Protein-making process:

- Messenger RNA (abbreviated mRNA) is a type of single-stranded RNA involved in protein synthesis.
- mRNA is made from a DNA template during the process of transcription.
- The role of mRNA is to carry protein information from the DNA in a cell's nucleus to the cell's cytoplasm (watery interior)
- In cytoplasm the protein-making machinery (ribosome) reads the mRNA sequence and translates each three-base codon into its corresponding amino acid in a growing protein chain



Miniproject 2: SSA vs. ODE solvers

Circadian rhythms and genetic oscillations: 16 reactions

$$
\begin{aligned}
A + R &\xrightarrow{\gamma_C} C \\
A &\xrightarrow{\delta_A} \emptyset \\
C &\xrightarrow{\delta_A} R \\
R &\xrightarrow{\delta_R} \emptyset \\
D_A + A &\xrightarrow{\gamma_A} D'_A \\
D_R + A &\xrightarrow{\gamma_R} D'_R
\end{aligned}
\qquad
\begin{aligned}
D'_A &\xrightarrow{\theta_A} A + D_A \\
D_A &\xrightarrow{\alpha_A} D_A + M_A \\
D'_A &\xrightarrow{\alpha'_A} D'_A + M_A \\
M_A &\xrightarrow{\delta_{MA}} \emptyset \\
M_A &\xrightarrow{\beta_A} A + M_A
\end{aligned}
\qquad
\begin{aligned}
D'_R &\xrightarrow{\theta_R} A + D_R \\
D_R &\xrightarrow{\alpha_R} D_R + M_R \\
D'_R &\xrightarrow{\alpha'_R} D'_r + M_R \\
M_R &\xrightarrow{\delta_{MR}} \emptyset \\
M_R &\xrightarrow{\beta_R} M_R + R
\end{aligned}
$$

Reproduce some figures of this article:
– José M. G. Vilar, Hao Yuan Kueh, Naama Barkai, Stanislas Leibler, *Mechanisms of noice-resistance in genetic oscillators*, PNAS April 30, 2002 vol. 99 no. 9 page 5988-5992.

Use either the course Python codes or the `GillesPy2` library (See Lab 1, part 4)