Implementation of an analytical-numerical approach to stochastization of one-step processes in the Julia programming language

Arseny V. Fedorov[1], Anna O. Masolova[1], Anna V. Korolkova[1] and Dmitry S. Kulyabov[1,2]

[1]*Peoples' Friendship University of Russia (RUDN University), 6 Miklukho-Maklaya St, Moscow, 117198, Russian Federation*

[2]*Joint Institute for Nuclear Research, 6 Joliot-Curie, Dubna, Moscow region, 141980, Russian Federation*

Abstract

In this paper, we use the method of stochastization of one-step processes to demonstrate the advantages of a stochastic representation of the system. To achieve this goal, the high-level scientific computing language Julia was chosen. Because the chosen approach is based on the paradigm of analytical-numerical calculations, then its implementation will be performed using auxiliary packages that are domain-specific extensions (DSL): Catalyst.jl and DifferentialEquations.jl. As a result of this work, the method of stochastization of one-step processes was applied to the SIR epidemic model.

Keywords

mathematic modeling, differential equations, schemes of kinetics reactions, stochastization of one-step processes, Julia programming language, DSL, Catalyst.jl, DifferentialEquations.jl

1. Introduction

The ordinary differential equations describe the deterministic behavior of the observed model which seldom does not represent the actual behavior of the system. Based on the previous research we suppose that stochastic differential equations can give better results.

One of the ways of obtaining SDE describing a system is to build self-consistent one-step processes. This method is based on the systems of interaction that allow for the opportunity to obtain SDE. This method is divided into two approaches: analytical (scheme of interaction, program code) and numerical (program code).

The goal of the research is the automatization of obtaining the equations and their solutions. For this purpose, the current task is solved in one computing environment.

Workshop on information technology and scientific computing in the framework of the XI International Conference Information and Telecommunication Technologies and Mathematical Modeling of High-Tech Systems (ITTMM-2021), Moscow, Russian, April 19–23, 2021

arsenyvitalievich@yandex.ru (A. V. Fedorov); anna.masolova@gmail.com (A. O. Masolova); korolkova-av@rudn.ru (A. V. Korolkova); kulyabov-ds@rudn.ru (D. S. Kulyabov)
https://yamadharma.github.io/ (D. S. Kulyabov)
0000-0003-1202-2787 (A. V. Fedorov); 0000-0002-3036-0117 (A. O. Masolova); 0000-0001-7141-7610 (A. V. Korolkova); 0000-0002-0877-7063 (D. S. Kulyabov)

The solution of this task is based on the paradigm of the analytical-numerical computations. For its realization, we chose the programming language Julia [1] which supports the creation of the subject-oriented plug-ins (DSL) [2]. Additionally, we are using the Catalyst.jl package which consists of the analytical and numerical modules, where the numerical one is based on the DifferentialEquations.jl plug-in.

The following algorithm had been developed: the entry point is the building of a set of kinetic reactions (based on the considered method of stochastization of one-step processes), describing the intercommunication of objects inside the model, then this set is interchanged into a scheme of interaction. Further, we conduct analytical computations, which interchange the obtained scheme into the SDE system. The SDE system is solved with the usage of the DifferentialEquations.jl packaging. As a result, we obtain graphs for the estimation of quality and quantity behavior of the system's characteristics.

This complex helps solve the problem of applying the analytical-numerical systems for obtaining the solution of our problems. Although it lacks the ability to obtain the SDE with a self-consistent stochastic part. The Catalyst.jl package needs upgrading by including program code in it to generate a self-consistent stochastic pat of the SDE.

2. Stochastization of the SIR model

In chemical kinetics, the reaction of interaction of substances can be represented as a time-dependent process. Equations are one of the ways to represent the dependences of the concentrations of substances. In this case, the differential equations can be represented in the form of kinetic reaction schemes. A complex of simple chemical reactions can be a complex reaction that is its composition.

There is a small amount of work in which the analytical approach is used in conjunction with the Julia language and the Catalyst.jl helper library used to model equations describing complex chemical reactions [3].

$$\begin{cases} \dfrac{dS}{dt} = \dfrac{-\beta SI}{N}, \\ \dfrac{dI}{dt} = \dfrac{\beta SI}{N} - \gamma I, \\ \dfrac{dR}{dt} = \gamma I, \end{cases} \tag{1}$$

where $S(t)$ is the number of susceptible individuals to the disease at the time t, $I(t)$ is the number of infected individuals at the time t, $R(t)$ is the number of affected individuals at the time t, $N = S(t) + I(t) + R(t)$, β is the coefficient of the intensity of interaction between healthy individuals and sick individuals (which subsequently leads to a disease), γ is the coefficient of the intensity of recovery of sick individuals.

On the example of the SIR epidemic model (1) [4, 5, 6, 7] demonstrate work Catalyst.jl package. This model was chosen due to its simplicity and frequent use in the field of mathematical modeling. It allows for the assessment and comparative analysis of the spread of the disease among individuals.

Based on the first equation of the system, we can say that the decrease in the number of susceptible individuals is directly proportional to the number of interactions with sick individuals. Those. a healthy individual becomes infected after direct contact.

From the second equation it follows that a decrease in the number of sick individuals is directly proportional to an increase in the number of recovered individuals. It is also worth noting that there is a similar relationship regarding the rate of disease of individuals due to the number of their contacts with susceptible ones.

And finally, the third equation shows a direct relationship between sick and recovered (i.e., the number of recovered is equal to the number of infected),

Summing up the description of the model, we can say that the process of spreading the epidemic consists of two stages:

1. Healthy individuals that are susceptible turn to ill;
2. Sick individuals who are was infected turn to recover.

$$S + I \xrightarrow{\beta} 2I, \tag{2}$$

$$I \xrightarrow{\gamma} R. \tag{3}$$

Now, for the model under consideration, we should write down the interaction scheme (2), (3). Then write out the chemical reactions included in the scheme according to the highlighted stages.

In the required format for implementation in Catalyst.jl, the diagram is written out as follows:

```
beta, S + I --> 2I
gamma, I --> R
```

Using the `@reaction_network` macro designed to process a set of reactions we set the interaction scheme, it is a complex chemical reaction (a set of chemical reactions):

```
using Catalyst
rn = @reaction_network begin
beta/N, S + I --> 2I
gamma, I --> R
end beta N gamma
```

Further when using a bundle of Jupyter Notebook and a graphical illustrator Graphviz we can visualize the recorded interaction scheme:

The Catalyst.jl package allows you to generate DE based on a recorded set of chemical reactions.

```
using DiffEqBase, OrdinaryDiffEq
u0    = [500.0, 10.0, 1.0]         # [S,I,R] at t=0
p     = [0.1, sum(u0), 0.01]       # [beta, gamma]
```

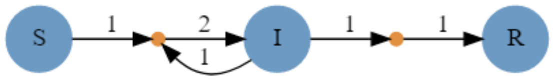

Figure 1: Interaction diagram for the SIR epidemic model

```
tspan = (0.0, 350.0)
op    = ODEProblem(rn, u0, tspan, p)
sol   = solve(op, RK4())            # use RK4 ODE solver
```

The Catalyst.jl module has a built-in ability to add stochastic to the model under consideration. First, a sampling algorithm is used, then a jump-like process is generated using the discretized equations, which is then solved.

This can be done with the following set of commands:

```
discrete_prob = DiscreteProblem(rn, u0, tspan, p)
jump_prob = JumpProblem(rn, discrete_prob, Direct())
sol = solve(jump_prob, SSAStepper())
```

However, this method does not allow us to obtain an SDE with a self-consistent stochastic part.

2.1. The one-step processes stochastization method

In order to obtain the desired stochastic form of the system describing the SIR epidemic model, we will apply the method of stochastization of one-step processes [8, 9, 10].

The first step is to introduce the vector ϕwhich is responsible for describing the states of the system, in our case $\phi = (S, I, R)$. Next, we introduce the operators $\overline{I}^{i}, \overline{F}^{i}$ (where i is the reaction number), denoting the initial and final states of the system, respectively, as well as the operator $\overline{R}^{i}$, obtained from their difference. Such operators are calculated using the following algorithm:

1. Compiled interaction diagram describing the flow of specific reactions in the system;
2. For the first reaction from the interaction scheme, the operators $\overline{I}^{1}$ и $\overline{F}^{1}$ will look like this (4):

$$\overline{I}^{1} = (1, 1, 0)^{T}, \quad \overline{F}^{1} = (0, 2, 0)^{T}. \tag{4}$$

To find these operators, write down the coefficient of the presence of parameters on the left side of the reaction. The form of writing the sought operators is based on the introduced vector of system states $\phi = (S, I, R)$, where the presence of each of the indicated parameters matters. The parameters S and I are present before the reaction occurs in a single copy, so we get the operator $\overline{I}^{1} = (1, 1, 0)^{T}$. Based on the right side of the reaction, we write out the operator $\overline{F}^{1}$where there is a coefficient 2 before the parameter I, which as a result forms $\overline{F}^{1} = (0, 2, 0)^{T}$.

3. For the second reaction, the operators $\overline{I}^2$ and $\overline{F}^2$ (5) are obtained in a similar way:

$$\overline{I}^2 = (0,1,0)^T, \quad \overline{F}^2 = (0,0,1)^T. \tag{5}$$

4. Then we introduce the state change operator R^i, obtained from the difference between the operators $\overline{F}^i$ and $\overline{I}^i$. For the first and second reactions, the operators $\overline{R}^1$, $\overline{F}^2$ (6) looks like this:

$$\overline{R}^1 = (-1,1,0)^T, \quad \overline{R}^2 = (0,1,-1)^T. \tag{6}$$

The next step in the implementation of this technique is to calculate the intensities of the transitions ${}^+S_{\underline{\alpha}}$, ${}^-S_{\underline{\alpha}}S$ (7), where i - is the reaction number, α is the number of reactions in the interaction scheme, the underlined index is the tensor component and $+/-$ — the direction of the reaction («+» — forward reaction, «−» — backward reaction):

The following are formulas for calculating the intensities of transitions (7) in general form:

$${}^+S_{\underline{\alpha}} = {}^+k_{\underline{\alpha}} \prod_{\underline{i}=1}^{n} \frac{\phi^{\underline{i}}!}{(\phi^{\underline{i}} - I^{\underline{i\alpha}})!}, {}^-S_{\underline{\alpha}} = {}^+k_{\underline{\alpha}} \prod_{\underline{i}=1}^{n} \frac{\phi^{\underline{i}}!}{(\phi^{\underline{i}} - F^{\underline{i\alpha}})!}. \tag{7}$$

Find the transition rates to our model, they are calculated separately for each of the reaction schemes of cooperation. Let us write down the intensity of the transition ${}^+S_1$ (8):

$${}^+S_1 = \beta * \frac{S!}{(S-1)!} * \frac{I!}{(I-1)!} = \beta * \frac{S*(S-1)!}{(S-1)!} * \frac{I*(I-1)!}{(I-1)!} = \beta SI. \tag{8}$$

The inverse intensity ${}^-S_1$ transition does not exist, since there is no backlash for our model. Similarly, we write out ${}^+S_2$ (9) (for this reaction the inverse intensity ${}^-S_2$ is also does not exist):

$${}^+S_2 = \gamma * \frac{I!}{(I-1)!} * \frac{R!}{R!} = \gamma I. \tag{9}$$

Obtaining the master kinetic equation (MKE) is an optional step, so let's move on to obtaining stochastic equations. Within the framework of this technique, they can be written in the Fokker-Planck form, which corresponds to the Langevin form. Obtaining the equation in the Fokker-Planck form is also an intermediate step; for brevity, in this work we will not write it out and write out the equation in the Langevin form right away. To do this, we need to calculate the drift vector A and the diffusion matrix B.

General formulas (10) for calculating A and B are presented below:

$$A(S,I,R) = {}^+S_{\underline{\alpha}} * R^{\underline{\alpha}}, \quad B(S,I,R) = {}^+S_{\underline{\alpha}} * R^{\underline{\alpha}} * (R^{\underline{\alpha}})^T. \tag{10}$$

Find A and B for our model:

$$A \;=\; \sum_{\alpha=1}^{2} {}^+S_{\underline{\alpha}} * R^{\underline{\alpha}} \;=\; {}^+S_1 * R^1 = \beta IS * (-1,1,0)^T + \gamma I * (0,-1,0)^T = \begin{pmatrix} -\beta IS \\ \beta IS - \gamma I \\ \gamma I \end{pmatrix}, \tag{11}$$

$$B = \sum_{\alpha=1}^{2} {}^{+}S_{\underline{\alpha}} * R^{\underline{\alpha}} * (R^{\underline{\alpha}})^T = {}^{+}S_1 * R^1 * (R^1)^T + {}^{+}S_2 * R^2 * (R^2)^T =$$

$$\beta IS * \begin{pmatrix} -1 \\ 1 \\ 0 \end{pmatrix} * (-1, 1, 0)^T + \gamma I * \begin{pmatrix} 0 \\ -1 \\ 1 \end{pmatrix} * (0, -1, 1)^T = \begin{pmatrix} \beta IS & -\beta IS & 0 \\ -\beta IS & \beta IS + \gamma I & -\gamma I \\ 0 & -\gamma I & \gamma I \end{pmatrix}. \quad (12)$$

Based on the obtained drift vector (11) and the diffusion matrix (12) we obtain an equation in the Langevin form (14):

$$dx(t) = A(\phi)dt + B(\phi)dW, \quad (13)$$

$$d\begin{pmatrix} S \\ I \\ R \end{pmatrix} = \begin{pmatrix} -\beta IS \\ \beta IS - \gamma I \\ \gamma I \end{pmatrix} dt + \begin{pmatrix} \beta IS & -\beta IS & 0 \\ -\beta IS & \beta IS + \gamma I & -\gamma I \\ 0 & -\gamma I & \gamma I \end{pmatrix} \begin{pmatrix} dW^1 \\ dW^2 \\ dW^3 \end{pmatrix}. \quad (14)$$

Thus, the process of stochastization of the equations of the SIR epidemic model was performed.

2.2. Obtaining solutions of the system of stochastic equations

The next step in this work will be to obtain a solution for the system of stochastic equations and to visualize them:

1. To do this, we will set additional system parameters:

```
beta = 0.1       #disease rate
gamma = 0.01     #recovery rate
N = sum(u0)
```

   Thus, a value for the disease rate β was set equal to 0.1. For the recovery rate value γ value was set to 0.05. Cofficient N is the sum of all the main parameters of the system — S, I, R.

2. Write the system of equations in the program form:

```
function sir(du, u, beta, gamma, N, t)
du[1] = -((beta*u[1]*u[2])/N)
du[2] = ((beta*u[1]*u[2])/N) - (gamma*u[2])
du[3] =  gamma*u[2]
end
```

   In the numerical approach, implemented using the DifferentialEquations.jl library, the model equations are specified as a function. It receives the following parameters as input:

   - du - array of technical parameters responsible for differentiation;
   - u - array of basic parameters described in the section ??;
   - beta, gamma, N - coefficients described above;

- t - time parameter;

3. Similarly, we declare a function that represents the stochastic part of the model:

```
function sir_noise(du, u, beta, gamma, N, t)
du[1] = sqrt(abs( ((beta*u[1]*u[2])/N)* (gamma*u[2]) ))
du[2] = sqrt(abs( ((beta*u[1]*u[2])/N)^2 +
 ↪  (((beta*u[1]*u[2])/N) +
(gamma*u[2]))^2 + (gamma*u[2])^2 ))
du[3] = sqrt(abs( ((beta*u[1]*u[2])/N)*(gamma*u[2]) ))
end
```

   At the input, it receives exactly the same parameters as the previous function. *sir*().

4. Obtain a solution:

```
prob = SDEProblem(sir, sir_noise, u0, tspan)
sol = solve(prob, EM())
```

   In order to solver *DifferentialEquations.jl* could correctly solve the problem you must specify the type of problem. In this case, for this we use the *SDEProblem*() function, which receives as input the deterministic form of our model (the *sir*()) function) and its stochastic form (the sir_noise() function), the initial conditions (*u*0) and the time interval (*tspan*) in which differentiation is performed. Next, we get a solution using the *solve*() function, to which the already transformed form of our stochastic model (*prob*) is fed to the input and a function that implements the solution algorithm is set, in this case we chose the Euler-Maruyama algorithm (function *EM*()).

This completes the solution of the system of stochastic equations.

2.3. Features of solving differential equations in the Julia programming language

In this part of the work, the subtleties that must be taken into account when working with differential equations in the Julia programming language will be described.

First of all, when working with differential equations, you need to use the DifferentialEquations.jl package, which contains the necessary tools for solving various kinds of equations. When it comes to obtaining a numerical solution, all the computational work is performed by the *solve*() function. For the most efficient construction of work with such a function, general options have been introduced that are compatible with any type of task. In this paper, we will point out some of the useful options:

- A standard set of hints for a solution algorithm, or rather a predefined vector *alg_hints*, which allows you to preset the type of problem.
  This is done with the following values: *stiff*, *nonstiff*, *auto*. The set value sets the type of the equation as rigid or non-rigid, respectively, the *auto* value allows the standard processing algorithm to connect additional procedures to determine the stiffness of the problem. The default value is *auto*.

Additional parameters are available for stochastic equations: : *additive* - denotes that the basic SDE has additive noise; : *stratonovich* - Indicates that the solution must be consistent with Stratonovich's interpretation.

- Output control is a set of arguments that control the settings of the solvers, according to which the numerical solution is derived. By default, the maximum set is used to provide the best user experience, but it can be scaled down to save the solution only at a finite point in time.
  The *saveat* parameter designates a specific time to save the numerical solution. The algorithm will store the values at each of the time points in the most efficient way available to the solver. For methods where interpolation is not possible, *saveat* can be equivalent to $tstops$.
  The $tstops$ parameter denotes the extra time that the time step algorithm must step over to help the solver handle discontinuities and singularities, since going exactly at the break will improve accuracy. If the method cannot change the time steps (multi-step methods with a fixed time step), then the time stops will use interpolation that matches the behavior of *saveat*. If the method cannot change time steps, and also cannot interpolate, then the value of $tstops$ must be a multiple of *dt*, otherwise an error will be thrown.
- Step dimension control is a set of arguments controlling procedures associated with setting the time step.
  The *dt* parameter sets the initial step size. It is also the step size for fixed time step methods. The value is selected automatically if the method is adaptive.
  The *dtmax* parameter sets the maximum value of the *dt* parameter for the adaptive time step. The default values are package dependent.
  The *dtmin* parameter sets the minimum value for the *dt* parameter for the adaptive time step. The default values are package dependent.

An additional part in the process of obtaining a numerical solution of equations is the selection of the most suitable algorithm. Some algorithms can give a more accurate solution, while others allow you to get it in a shorter time, but with a more significant error. As an overview, let us designate some of the possible algorithms for application to the types of problems demonstrated in our work.

Speaking about the solving of the ODE system describing the deterministic form of the system, we can consider the following algorithms for its solution:

- Euler: Direct Euler method with fixed variable step.;
- Midpoint - second order midpoint method. Uses built-in Euler method for adaptability;
- Heun: second-order Heun's method, Euler's method is used in a similar way;
- Ralston - Optimized second-order midpoint method, similarly used Euler's method;
- RK4: Runge–Kutta method of the fourth order. An error control technique is used for an adaptive step with a maximum error over the entire interval;
- BS3: Bogatsky-Champin method 3/2.

When reducing our ODE to the form of a jump-like process, we are able to solve such a system using the following algorithm:

- SSAStepper: step-by-step algorithm for pure problems like ConstantRateJump and / or MassActionJump. It supports DiscreteCallback handling and persistence control options such as saveat.

To solve the SDE system, we can use the following algorithms:

- EM: Euler-Maruyama method. Has a strict order of 0.5 in Ito terms. It can handle all kinds of noise, including off-diagonal, scalar and color noise. The peculiarity of the algorithm is that for its use it is necessary to set only a fixed time step;
- EulerHeun: Euler-Heun method. Strong order equal to 0.5 in terms of Stratonovich. It can handle all kinds of noise, including off-diagonal, scalar and color noise. Fixed time step only;
- LambaEM: modified Euler-Maruyama method with adaptive time step with error estimation, research authors Lambe and Rakkaukas. Can handle all kinds of noise, including off-diagonal, scalar and color noise;
- SRI: the algorithm has a strict order of 1.5 for Ito's diagonal / scalar SDEs. The backend is based on the SRIW1 method; item SRIW1 - adaptive strong order 1.5 and weak order 2.0 for diagonal / scalar SDEs in Ito form.

3. Results

Before proceeding to the descriptive part of the materials (graphs) that were obtained in the course of this research work, we want to note that for the deterministic case describing the epidemic model, solution graphs were immediately built without any modification. To obtain graphs describing the behavior of the epidemic model with the addition of discrete stochastic, the kinetic Monte Carlo method (Gillepsy algorithm) built into the DifferentialEquations.jl library was used. And finally, in order to obtain a graph describing the most realistic variant of the model's behavior (with the addition of self-consistent stochastic), the method of stochastization of one-step processes was used. Thus, in the course of this work, the graphs of the solution for the system of equations of the SIR model were obtained:

- In the deterministic form (Fig. 2);
- With the addition of discrete stochastic (Fig. 3);
- With the addition of self-consistent stochastic (Fig. 4);
- General graph comparing all solutions (Fig. 5).

Graph describing the deterministic case solution to the epidemic model (Fig. 2) has a sleek look. This is due to the fact that the system does not have any impacts on the model (external / internal).

Judging by the resulting graph, it can be seen that the number of susceptible individuals (parameter $S(t)$ marked as the blue curve) is rapidly falling and reaches zero by the time of the model time equal to 125 seconds. At the same time, the number of infected individuals (parameter $I(t)$ marked as the red dashed curve) in the same rapid manner begins to increase until the number of susceptible individuals has a value close to zero. This moment comes

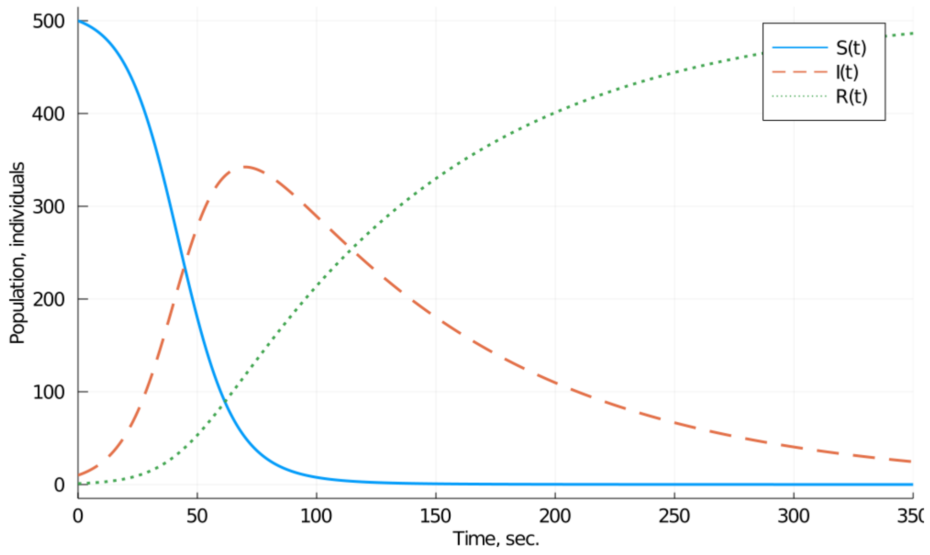

Figure 2: Deterministic system solution graph for the SIR model

when the value of the model time approaches 100 seconds. The curve responsible for the designation of the number of affected individuals (parameter $R(t)$ marked as the green dashed curve) increases throughout the model time, dynamically changing the growth rate. The ratio of the growth rate of recovered individuals is associated with the onset of the peak value on the graph of the curve indicating the number of infected individuals (75 seconds of model time). As soon as the number of infected individuals approaches the maximum value and stops growing, the number of infected individuals begins to grow rapidly. Accordingly, when the number of infected individuals approaches zero, the growth dynamics of the curve denoting those who have recovered falls.

Graph describing the solution of the model with the addition of discrete stochastic (Fig. 3) has a similar nature of behavior, however, there are minor fluctuations. This is due to the fact that an impact is added to the system based on a discrete stepwise process.

Graph describing the solution of the stochastized model using the method of stochastization of one-step processes(Fig. 4) has more pronounced fluctuations in comparison with the two previous cases.

This is due to the fact that a term with a self-consistent stochastic is added, which is responsible for external and internal influences. This approach requires clarification: due to the added stochastics, cases of the appearance of new individuals in the system from the vacuum, as well as their disappearance, are possible. Described behavior of model characteristics ($S(t)$, $I(t)$, $R(t)$) can be seen on the graph at the time interval from 50 to 100 seconds of model time.

There is also a general graph comparing all solutions. (Fig. 5).

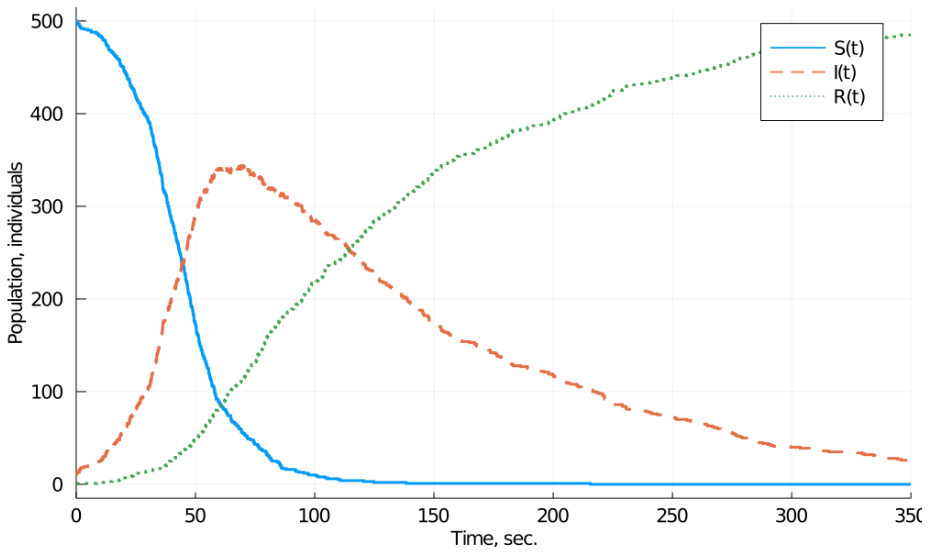

Figure 3: System solution graph with the addition of discrete stochastic for the SIR model

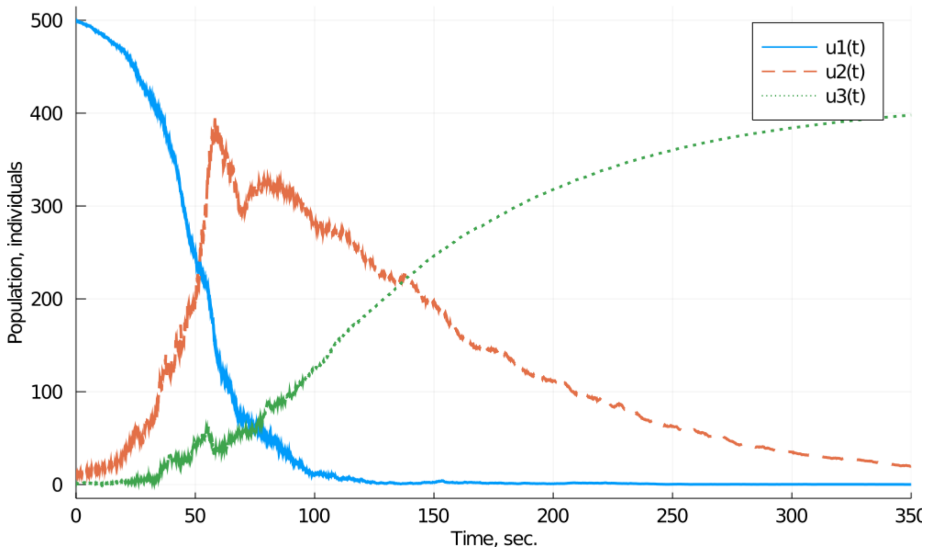

Figure 4: System solution graph with the addition of self-consistent stochastic for the SIR model

4. Conclusion

In the course of this work, the method of stochastization of one-step processes was applied to the SIR epidemic model.

A software implementation of the computational part for obtaining a solution to the ODE system has been developed. The broad capabilities of the high-level programming language Julia are demonstrated when working with differential equations of different types.

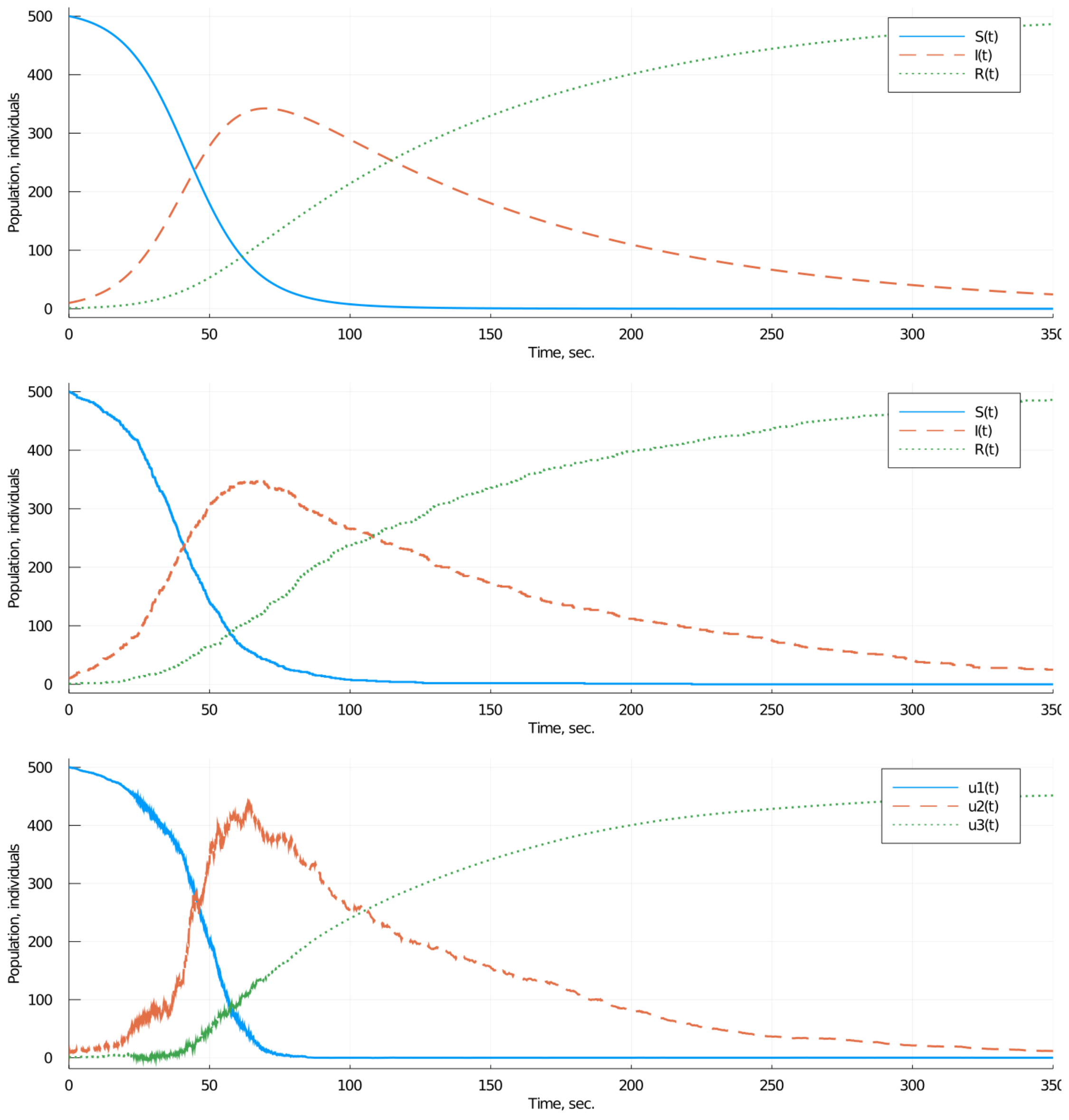

Figure 5: General graph of the three solutions for the SIR model

A solution is obtained for each type of model, graphs are constructed that describe the main probabilistic-temporal characteristics of the model, and a numerical analysis of each of the model variants is performed.

Acknowledgments

This paper has been supported by the RUDN University Strategic Academic Leadership Program and by Russian Foundation for Basic Research (RFBR) according to the research project No 19-01-00645.

References

[1] J. Bezanson, S. Karpinski, V. B. Shah, A. Edelman, Julia: A fast dynamic language for technical computing (2012) 1–27. arXiv:1209.5145.

[2] D. S. Kulyabov, A. V. Korol'kova, Computer algebra in julia, Programming and Computer Software 47 (2021) 133–138. doi:10.1134/S0361768821020079. arXiv:2108.12301.

[3] A. V. Fedorov, A. O. Masolova, A. V. Korolkova, D. S. Kulyabov, Application of a numerical-analytical approach in the process of modeling differential equations in the julia language, in: Information Technology, Telecommunications and Control Systems (ITTCS), 2020, volume 1694 of *Journal of Physics: Conference Series*, IOP Publishing, Innopolis, 2020, pp. 012026.1–8. doi:10.1088/1742-6596/1694/1/012026.

[4] R. Ross, An application of the theory of probabilities to the study of a priori pathometry.—part i, Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character 92 (1916) 204–230. doi:10.1098/rspa.1916.0007.

[5] R. Ross, An application of the theory of probabilities to the study of a priori pathometry.—part ii, Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character 93 (1917) 212–225. doi:10.1098/rspa.1917.0014.

[6] R. Ross, An application of the theory of probabilities to the study of a priori pathometry.—part iii, Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character 93 (1917) 225–240. doi:10.1098/rspa.1917.0015.

[7] W. O. Kermack, A. G. McKendrick, A contribution to the mathematical theory of epidemics, Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character 115 (1927) 700–721. doi:10.1098/rspa.1927.0118.

[8] A. V. Korolkova, D. S. Kulyabov, One-step stochastization methods for open systems, EPJ Web of Conferences 226 (2020) 02014.1–4. doi:10.1051/epjconf/202022602014.

[9] M. Hnatič, E. G. Eferina, A. V. Korolkova, D. S. Kulyabov, L. A. Sevastyanov, Operator approach to the master equation for the one-step process, EPJ Web of Conferences 108 (2016) 02027. doi:10.1051/epjconf/201610802027. arXiv:1603.02205.

[10] E. G. Eferina, M. Hnatich, A. V. Korolkova, D. S. Kulyabov, L. A. Sevastianov, T. R. Velieva, Diagram representation for the stochastization of single-step processes, in: V. M. Vishnevskiy, K. E. Samouylov, D. V. Kozyrev (Eds.), Distributed Computer and Communication Networks. DCCN 2016. Communications in Computer and Information Science, volume 678 of *Communications in Computer and Information Science*, Springer International Publishing, Cham, 2016, pp. 483–497. doi:10.1007/978-3-319-51917-3_42.