Construction and simulation of continuous time portfolio with given properties

Sergey G. Shorokhov[1]

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

Abstract

A continuous-time version of portfolio management problem for a market with multiple stocks is under study. Asset allocation policy is determined implicitly as a solution to the system of ordinary differential equations for asset quantities. Right-hand sides of equations are derived explicitly in closed form from given portfolio properties using techniques of construction of differential equations by given integral manifold. Related issues of simulation for portfolio with desired properties are also addressed. We present the results of computer experiment for verification of specified portfolio property using constructed portfolio model in the form of the system of stochastic differential equations.

Keywords

Markov decision process, portfolio policy, stochastic process, SDE simulation, OpenCL framework

1. Introduction

Modeling with stochastic differential equations is used extensively in many areas of modern research [1]. Stochastic differential equations (SDE) arise in modeling and simulation of various random dynamical systems in physical [2], chemical [3], biological [4], financial [4] and other sciences.

Stochastic models based on SDE are an integral part of quantitative analysis of wireless channels in communications [5]. Modeling of signals and interference in communications requires construction of SDE with specified properties, such as given marginal probability density function and autocovariance function. Applications of SDE in other areas usually also imply construction of SDE with desired characteristics.

We study asset allocation problem for a portfolio with specified properties using continuous-time continuous-state Markov decision process, determined implicitly by a system of differential equations. Our goal is to derive right-hand sides of equations by known portfolio properties and verify the required portfolio property in a computer experiment using simulation techniques.

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

shorokhov-sg@rudn.ru (S. G. Shorokhov)
https://esystem.rudn.ru/users/3295 (S. G. Shorokhov)
0000-0001-6835-4110 (S. G. Shorokhov)

2. Implicit Markov decision process for portfolio management

Markov decision process [6] is a widely used approach to decision making, including applications in finance.

Consider a market in which n no dividend stocks are traded continuously and transaction cost and consumption are not taken into account. The stock price processes $s_i(t)$, $i = \overline{1,n}$ generally satisfy the system of SDE [7]:

$$ds_i = \mu_i(t, \boldsymbol{s})\, dt + \sum_{j=1}^{m} \sigma_{ij}(t, \boldsymbol{s})\, dW_j,\ i = \overline{1,n},\ m \leqslant n, \tag{1}$$

where $\mu_i(t, \boldsymbol{s})$ are drift terms, $\sigma_{ij}(t, \boldsymbol{s})$ are volatility terms, $W_1, ..., W_m$ are independent standard Wiener processes, $\boldsymbol{s} = (s_1, ..., s_n)$.

The initial stock prices at $t = t_0$ are assumed to be known:

$$s_i(t_0) = s_i^{(0)} > 0,\ i = \overline{1,n}. \tag{2}$$

Let $x_i(t)$ be the quantity (in units) of the i-th stock in the portfolio at time t. We assume that the quantity $x_i(t)$ can be positive (long position), negative (short position) or zero (no position) and can take fractional values.

The value process of the stock portfolio $P(t)$ at time t is determined by the formula

$$P(t) = \sum_{i=1}^{n} x_i(t)\ s_i(t). \tag{3}$$

The share (weight) $w_i(t)$ of the i-th stock in the portfolio $P(t)$ is equal to

$$w_i(t) = \frac{x_i(t)\ s_i(t)}{P(t)} = \frac{x_i(t)\ s_i(t)}{\sum_{j=1}^{n} x_j(t)\ s_j(t)},\ i = \overline{1,n}. \tag{4}$$

The weights $w_i(t)$ are not all independent, because of the norming condition $\sum_{i=1}^{n} w_i(t) = 1$.

The state of the stock portfolio can be determined either by prices $\mathbf{s} = (s_1, ..., s_n)$ and quantities $\mathbf{x} = (x_1, ..., x_n)$, or by prices $\mathbf{s} = (s_1, ..., s_n)$, weights $\mathbf{w} = (w_1, ..., w_n)$ and total portfolio value $P(t)$. We assume that the portfolio state is given by price-quantity pair $(\mathbf{s}, \mathbf{x})$.

The portfolio state $(\mathbf{s}, \mathbf{x})$ varies because of changes in stock prices $\mathbf{s}$ due to market situation or changes in stock quantities $\mathbf{x}$ due to policy (actions) of portfolio manager.

Possible actions for the i-th stock in the portfolio are to hold the stock, to buy or to sell some units of the stock.

As a result of actions for the time period $\triangle t > 0$ change in the quantity of units of the i-th stock in the portfolio is equal to $\triangle x_i = x_i(t + \triangle t) - x_i(t)$. If $\triangle x_i > 0$, then $\triangle x_i$ units of the stock are purchased, if $\triangle x_i < 0$, then $|\triangle x_i|$ units of the stock are sold, if $\triangle x_i = 0$, then $x_i(t)$ units of the stock are being held in the portfolio.

Asset allocation policy $\mathbf{x}(t)$ may be explicit with quantities of stocks being functions of time and/or stock prices, i.e. $\mathbf{x} = \mathbf{x}(t, \mathbf{s})$. We assume that asset allocation policy $\mathbf{x}(t)$ is determined implicitly as a solution of the system of differential equations

$$dx_i = f_i(t, \mathbf{s}, \mathbf{x})\, dt,\ i = \overline{1,n} \tag{5}$$

with initial conditions (initial stock quantities)

$$x_i(t_0) = x_i^{(0)},\ i = \overline{1,n}. \tag{6}$$

Actually, the system (5) should be treated as a system of SDE with zero diffusion coefficients, because right-hand sides of system (5) depend on stochastic stock price processes $s_i(t)$, following (1) with initial conditions (2).

When functions $f_i(t, \mathbf{s}, \mathbf{x})$ in (5) are fixed, the portfolio management policy is fully determined by equations (1) and (5) with initial conditions (2) and (6), the portfolio dynamics depends only on the initial state of the portfolio $\left(\mathbf{s}^{(0)},\ \mathbf{x}^{(0)}\right)$ at time t_0 and is independent of the past ($t < t_0$).

Thus, we determine the portfolio policy by choosing various functions $f_i(t, \mathbf{s}, \mathbf{x})$ on the right-hand side of equations (5) and receive Markov decision process for portfolio management.

3. Construction of portfolio with given properties

Construction of stock portfolio with desired characteristics and evaluation of its risk and performance using simulation and optimization plays extremely important role in modern financial industry [8]. The desired properties of the portfolio can be a given rate of return, a given risk metric (variance of return, value at risk, expected shortfall), given structural (country, industry, rating) ratios, etc.

As we will show below, portfolio management policy may be based on construction of ordinary differential equations (ODE) by a given integral manifold. Originally, construction of ODE from a given integral curve was proposed by Yerugin [9] and later extended to dynamical systems of general nature by Galiullin [10], Mukharlyamov [11] and many other authors [12].

The method is widely used in investigations of controlled dynamical systems and allows to construct equations of motion from given properties of trajectories taking into account additional requirements, such as stability of the given manifold or optimality in some sense. The problem of construction of equations in the class of Ito SDE by known properties of motion was investigated by Tleubergenov [13, 14, 15].

We assume that price dynamics of stocks in the portfolio is described by SDE (1) with initial conditions (2) and quantities of stocks follow Markov portfolio policy, implicitly determined by equations (5) with initial conditions (6). The functions on the right-hand sides of equations (1) and (5) are assumed to be continuous in time t and Lipschitz in portfolio state variables $\mathbf{s}, \mathbf{x}$.

We consider portfolio properties in the form of l equalities

$$\omega_k(t,\ \mathbf{s},\ \mathbf{x}) = 0,\ k = \overline{1,\ l},\ l < n \tag{7}$$

where Jacobian $l \times n$-matrix $\left(\frac{\partial}{\partial \mathbf{x}}\right)$ is of maximum rank l.

Basically, the stochasticity of stock prices driven by SDE (1) leads to violation of equalities (7), so our goal is to find a portfolio policy of the form (5), which ensures that equalities (7) are satisfied on ensemble average, i.e.

$$\mathbb{E}\left[\omega_k(t,\ \mathbf{s}(t),\ \mathbf{x}(t))\right] = 0,\ k = \overline{1,\ l},\ \forall t \geqslant t_0, \tag{8}$$

where $\mathbb{E}$ denotes expected value (ensemble average) [16] and for a stochastic process $X(t)$

$$\mathbb{E}\left[X(t)\right] = \int\limits_{-\infty}^{+\infty} x f_X(x,\, t)\, dx, \tag{9}$$

where $f_X(x,\, t)$ is the probability density function of $X(t)$ at time t.

This statement of the problem differs from common approach to construction of stochastic differential equations by given integral manifold [13], when properties (7) are supposed to be satisfied exactly. But in portfolio management problem this implies restrictions on price equations (1), which is not relevant for financial market models.

Stochastic differentials of ω_k can be calculated using the multidimensional Ito's formula [17]

$$d\omega_k = \left(\frac{\partial \omega_k}{\partial t} + \sum_{i=1}^{n} \frac{\partial \omega_k}{\partial s_i} \mu_i + \sum_{i=1}^{n} \frac{\partial \omega_k}{\partial x_i} f_i + \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{h=1}^{m} \sigma_{ih} \sigma_{jh} \frac{\partial^2 \omega_k}{\partial s_i \partial s_j} \right) dt + \\ + \sum_{i=1}^{n} \sum_{h=1}^{m} \sigma_{ih} \frac{\partial \omega_k}{\partial x_i} dW_h,\ k = \overline{1,\, l}. \tag{10}$$

Arbitrary (unknown) functions f_i in (5) can be chosen in such a way that the following equalities hold true

$$\frac{\partial \omega_k}{\partial t} + \sum_{i=1}^{n} \frac{\partial \omega_k}{\partial s_i} \mu_i + \sum_{i=1}^{n} \frac{\partial \omega_k}{\partial x_i} f_i + \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{h=1}^{m} \sigma_{ih} \sigma_{jh} \frac{\partial^2 \omega_k}{\partial s_i \partial s_j} = - \sum_{q=1}^{l} \lambda_{kq} \omega_q, k = \overline{1,\, l}, \tag{11}$$

where $\lambda_{kq}(t, \mathbf{s}, \mathbf{x})$ are arbitrary functions. Equations (11) can be regarded as a system of l linear algebraic equations for n unknown functions f_i:

$$\sum_{i=1}^{n} \frac{\partial \omega_k}{\partial x_i} f_i = -\varphi_k,\ k = \overline{1,\, l}, \tag{12}$$

where

$$\varphi_k = \sum_{q=1}^{l} \lambda_{kq} \omega_q + \frac{\partial \omega_k}{\partial t} + \sum_{i=1}^{n} \frac{\partial \omega_k}{\partial s_i} \mu_i + \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{h=1}^{m} \sigma_{ih} \sigma_{jh} \frac{\partial^2 \omega_k}{\partial s_i \partial s_j}.$$

Applying the methodology of Moore–Penrose pseudoinverse matrices [18] to system (12), we obtain the solution of (12) in the following matrix form:

$$\mathbf{f} = -\left(\frac{\partial}{\partial \mathbf{x}}\right)^{+} + \left[\mathbb{I}_n - \left(\frac{\partial}{\partial \mathbf{x}}\right)^{+} \frac{\partial}{\partial \mathbf{x}}\right] \Phi, \left(\frac{\partial}{\partial \mathbf{x}}\right)^{+} = \frac{\partial}{\partial \mathbf{x}}^{T} \left(\frac{\partial}{\partial \mathbf{x}} \frac{\partial}{\partial \mathbf{x}}^{T}\right)^{-1}, \tag{13}$$

where $\phi = (\varphi_1, ..., \varphi_l)^T$, $\mathbb{I}_n$ is the identity $n \times n$-matrix, $\Phi = (\Phi_1, ..., \Phi_n)$ is a vector of arbitrary functions. The pseudoinverse matrix $\left(\frac{\partial}{\partial \mathbf{x}}\right)^{+}$ exists, because the Jacobian matrix $\frac{\partial}{\partial \mathbf{x}}$ has the highest rank l. Substituting the corresponding expression from (12) for ϕ, we obtain the solution of (12) in the following final form

$$\mathbf{f} = -\left(\frac{\partial}{\partial \mathbf{x}}\right)^{+} \left[\Lambda\, \omega + \frac{\partial}{\partial t} + \frac{\partial}{\partial \mathbf{s}} \mu + \frac{1}{2} \mathbf{tr}\left(\boldsymbol{\sigma}^T \frac{\partial^2}{\partial \mathbf{x}^2} \boldsymbol{\sigma}\right)\right] + \left[\mathbb{I}_n - \left(\frac{\partial}{\partial \mathbf{x}}\right)^{+} \frac{\partial}{\partial \mathbf{x}}\right] \Phi, \tag{14}$$

where $\Lambda = \left(\lambda_{kq}\right)$, **tr** is matrix trace operation.

We additionally assume that for any $k, q = \overline{1, l}$ the random variables λ_{kq} and ω_q are independent, then, averaging the equalities (10) along ensemble of trajectories under the policy (14), we receive that

$$\frac{d\mathbb{E}\left[\omega_k\right]}{dt} = -\sum_{q=1}^{l} \mathbb{E}\left[\lambda_{kq}\right] \mathbb{E}\left[\omega_q\right], \; k = \overline{1, l}. \tag{15}$$

The system of ODE (15) admits the trivial solution $\mathbb{E}\left[\omega_k\right] = 0$, $k = \overline{1, l}$, which implies that the portfolio with stock dynamics (1) and portfolio policy (5) admits portfolio properties (8).

Thus, we receive the following statement on Markov portfolio policies with given portfolio properties.

Markov asset allocation policy (5) *with right-hand sides given by* (14) *admits the specified portfolio properties* (8)*, provided that* $\Lambda = \left(\lambda_{kq}\left(t, \mathbf{s}, \mathbf{x}\right)\right)$ *is an arbitrary* $l \times l$*-matrix such that the random variables* $\lambda_{kq}\left(t,\ \mathbf{s}\left(t\right),\ \mathbf{x}\left(t\right)\right)$ *and* $\omega_q\left(t,\ \mathbf{s}\left(t\right),\ \mathbf{x}\left(t\right)\right)$ *are independent,* $\left(t, \mathbf{s}, \mathbf{x}\right)$ *is an arbitrary column vector.*

Portfolio asset allocation policy (5) with right-hand sides given by (14) can be applied to portfolios containing stocks, currencies and commodities. By choosing arbitrary functions λ_{kq} and Φ_i it is possible to ensure the stability of portfolio policy or to find an optimal policy for given reward function and discount factor.

Asset portfolio can include cash account, be self-financing, and incorporate other additional features. The value of portfolio with cash account and n risky assets is equal to

$$P(t) = x_0\left(t\right) + \sum_{i=1}^{n} x_i\left(t\right)\ s_i\left(t\right),$$

where $x_0\left(t\right)$ denotes the balance of cash account at time t.

The portfolio is self-financing [17], if no external inflow or outflow of cash and stocks takes place and the purchase of a stock must be financed by cash on cash account or by sale of some stocks from the portfolio. The portfolio with cash account is self-financing, if [17]

$$dP = \sum_{i=1}^{n} x_i ds_i + r x_0 dt,$$

where $r > 0$ is the fixed interest rate of cash account. This implies that the balance of cash account x_0 satisfies the following equation [19]

$$dx_0 = \left(r x_0 - \sum_{i=1}^{n} s_i f_i\right) dt,$$

where functions f_i from (5) determine the portfolio policy.

4. Modeling self-financing portfolio with given structure

We consider a self-financing portfolio with cash account and two risky assets (stocks), driven by geometric Brownian motion, and assume that the portfolio management policy is given in

implicit form (5). The portfolio dynamics is determined by the following system of SDE:

$$\begin{cases} ds_1 = \mu_1 s_1\, dt + \sigma_1 s_1\, dW_1, \\ ds_2 = \mu_2 s_2\, dt + \sigma_2 s_2\, dW_2, \\ dx_0 = (r x_0 - s_1 f_1(t,\ \mathbf{s},\ \mathbf{x}) - s_2 f_2(t,\ \mathbf{s},\ \mathbf{x}))\, dt, \\ dx_1 = f_1(t,\ \mathbf{s},\ \mathbf{x})\, dt, \\ dx_2 = f_2(t,\ \mathbf{s},\ \mathbf{x})\, dt. \end{cases} \tag{16}$$

Here s_1, s_2 are the stock prices, x_0 is the balance of cash account, x_1, x_2 are the quantities of stocks in the portfolio, $\mathbf{s} = (s_1, s_2)$, $\mathbf{x} = (x_0, x_1, x_2)$, μ_1, μ_2 are the instantaneous rates of return, σ_1, σ_2 are the volatilities of stocks, W_1, W_2 are independent standard Wiener processes, r is the interest rate of cash account.

We assume that the desired portfolio property is the condition that 80% of the portfolio assets is invested in risky assets (stocks).

The value of the portfolio under consideration is equal to $P = x_0 + x_1 s_1 + x_2 s_2$, and weights of cash and stocks are equal to

$$w_0 = \frac{x_0}{x_0 + x_1 s_1 + x_2 s_2},\ w_1 = \frac{x_1 s_1}{x_0 + x_1 s_1 + x_2 s_2},\ w_2 = \frac{x_2 s_2}{x_0 + x_1 s_1 + x_2 s_2}.$$

The desired portfolio property can be expressed as the following relation between the stock weights w_1 and w_2:

$$w_1 + w_2 = \frac{x_1 s_1 + x_2 s_2}{x_0 + x_1 s_1 + x_2 s_2} = 0.8,$$

which can be transformed into the equality

$$\omega \equiv -4x_0 + x_1 s_1 + x_2 s_2 = 0. \tag{17}$$

Our goal is to determine the unknown functions f_1, f_2 in (16) to guarantee the portfolio property (17) on ensemble average.

According to Ito's lemma the stochastic differential of ω is equal to

$$d\omega = (-4r x_0 + \mu_1 x_1 s_1 + \mu_2 x_2 s_2 + 5 s_1 f_1 + 5 s_2 f_2)\, dt + \sigma_1 s_1^2 dW_1 + \sigma_2 s_2^2 dW_2. \tag{18}$$

We set the unknown functions f_1, f_2 so, that the drift term at dt is equal to $-\alpha\omega$, $\alpha \in \mathbb{R}$:

$$-4r x_0 + \mu_1 x_1 s_1 + \mu_2 x_2 s_2 + 5 s_1 f_1 + 5 s_2 f_2 = -\alpha\omega. \tag{19}$$

Equality (19) is the linear algebraic equation for the determination of unknown functions f_1, f_2, and the general solution of equation (19) is:

$$f_1 = -\frac{s_1}{s_1^2 + s_2^2}\varphi + s_2\psi,\ f_2 = -\frac{s_2}{s_1^2 + s_2^2}\varphi - s_1\psi, \tag{20}$$

where $\varphi = 0.2\alpha\omega - 0.8 r x_0 + 0.2\mu_1 x_1 s_1 + 0.2\mu_2 x_2 s_2$, $\alpha \in \mathbb{R}$, ψ is an arbitrary function of $t, \mathbf{s}, \mathbf{x}$.

Thus, SDE system (16) for modeling prices and quantities of the portfolio with given property (17) takes the following form with an arbitrary constant $\alpha \in \mathbb{R}$ and an arbitrary function ψ:

$$\begin{cases} ds_1 = \mu_1 s_1\, dt + \sigma_1 s_1\, dW_1, \\ ds_2 = \mu_2 s_2\, dt + \sigma_2 s_2\, dW_2, \\ dx_0 = (\alpha\omega + 0.2rx_0 + 0.2\mu_1 x_1 s_1 + 0.2\mu_2 x_2 s_2)\, dt, \\ dx_1 = \left(-\dfrac{s_1}{s_1^2 + s_2^2}\,(0.2\alpha\omega - 0.8rx_0 + 0.2\mu_1 x_1 s_1 + 0.2\mu_2 x_2 s_2) + s_2\psi\right) dt, \\ dx_2 = \left(-\dfrac{s_1}{s_1^2 + s_2^2}\,(0.2\alpha\omega - 0.8rx_0 + 0.2\mu_1 x_1 s_1 + 0.2\mu_2 x_2 s_2) - s_1\psi\right) dt. \end{cases} \tag{21}$$

5. Simulation of self-financing portfolio with given structure

To verify that the portfolio model (21) satisfies the portfolio property (17) we simulate the trajectories of system (21) for the time segment $[t_0, T]$.

So we divide the segment $[t_0, T]$ into N equal parts of length $h = \frac{T - t_0}{N}$ and simulate a discretized version of SDE (21).

There exists a number of discretization schemes available, such as Milstein scheme [20] and stochastic version of Runge-Kutta methods [21, 22], but the most intuitive, easy to implement and common is Euler-Maruyama scheme [23]. Euler-Maruyama discretization of (21) gives us the following equations:

$$\begin{cases} s_{1,k+1} = s_{1,k} + \mu_1 s_{1,k} h + \sigma_1 s_{1,k}\sqrt{h} Z_{1,k}, \\ s_{2,k+1} = s_{2,k} + \mu_2 s_{2,k} h + \sigma_2 s_{2,k}\sqrt{h} Z_{2,k}, \\ x_{0,k+1} = x_{0,k} + \left(\alpha\omega_k + 0.2rx_{0,k} + 0.2\mu_1 x_{1,k} s_{1,k} + 0.2\mu_2 x_{2,k} s_{2,k}\right) h, \\ x_{1,k+1} = x_{1,k} - \dfrac{0.2\alpha\omega_k - 0.8rx_{0,k} + 0.2\mu_1 x_{1,k} s_{1,k} + 0.2\mu_2 x_{2,k} s_{2,k}}{s_{1,k}^2 + s_{2,k}^2} s_{1,k} h + s_{2,k}\psi h, \\ x_{2,k+1} = x_{2,k} - \dfrac{0.2\alpha\omega_k - 0.8rx_{0,k} + 0.2\mu_1 x_{1,k} s_{1,k} + 0.2\mu_2 x_{2,k} s_{2,k}}{s_{1,k}^2 + s_{2,k}^2} s_{2,k} h - s_{1,k}\psi h, \end{cases} \tag{22}$$

where $s_{i,k} = s_i(t_k)$, $x_{i,k} = x_i(t_k)$, $t_k = t_0 + kh$, $\omega_k = -4x_{0,k} + x_{1,k}s_{1,k} + x_{2,k}s_{2,k}$, $Z_{i,k}$ are i.i.d. random variables with standard normal distribution, i.e. $Z_{i,k} \sim \mathcal{N}(0, 1)$.

Basically, simulation of SDE is a resource-intensive application, so to speed up the simulation one has to enable GPU acceleration. PyOpenCL [24] is a programming environment for access to OpenCL parallel computation framework [25] from Python language.

To apply PyOpenCL environment for portfolio simulation, one has to follow the typical sequence of steps:

1. Import the OpenCL API, obtain an OpenCL platform and a GPU device id:

```
import pyopencl as cl
plat = cl.get_platforms()[0]
dev = plat.get_devices()[2]
```

2. Initialize a context for the selected GPU device, create a Queue object (with profiling enabled to track computation time), create memory buffers on GPU for input and output data (here p, q and s_gpu are NumPy arrays):

```
opencl_context = cl.Context(devices=[dev])
command_queue = cl.CommandQueue(opencl_context,
    properties=cl.command_queue_properties.PROFILING_ENABLE)
p_buffer = cl.Buffer(opencl_context,
    cl.mem_flags.READ_ONLY | cl.mem_flags.COPY_HOST_PTR, hostbuf=p)
q_buffer = cl.Buffer(opencl_context,
    cl.mem_flags.READ_ONLY | cl.mem_flags.COPY_HOST_PTR, hostbuf=q)
s_buffer = cl.Buffer(opencl_context,
    cl.mem_flags.WRITE_ONLY, s_gpu.nbytes)
```

3. Store the source code of C functions for GPU execution in Python strings (`sde_step_src` contains source code of one step of SDE simulation according to (22), `simulate_sde_src` contains source code of main kernel function, being called from Python):

```
sde_step_src = """
// One step of SDE simulation
static void sde_step(__global double *p, double *x, double W)
{
double s1n, s2n, x0n, x1n, x2n;
x[5] = -0.8*x[2] + 0.2*x[3]*x[0] + 0.2*x[4]*x[1]; // omega
s1n = p[P_M1]*x[0]*p[P_DT] + p[P_S1]*x[0]*p[P_SQRDT]*W;
s2n = p[P_M2]*x[1]*p[P_DT] + p[P_S2]*x[1]*p[P_SQRDT]*W;
x0n = (p[P_A]*x[5] + 0.2*p[P_R]*x[2] + 0.2*p[P_M1]*x[3]*x[0] +
     0.2*p[P_M2]*x[4]*x[1])*p[P_DT];
x1n = -x[0]*(p[P_A]*x[5] - 0.8*p[P_R]*x[2] + 0.2*p[P_M1]*x[3]*x[0] +
     0.2*p[P_M2]*x[4]*x[1])*p[P_DT]/(x[0]*x[0]+x[1]*x[1]);
x2n = -x[1]*(p[P_A]*x[5] - 0.8*p[P_R]*x[2] + 0.2*p[P_M1]*x[3]*x[0] +
     0.2*p[P_M2]*x[4]*x[1])*p[P_DT]/(x[0]*x[0]+x[1]*x[1]);
x[0] += s1n; x[1] += s2n; x[2] += x0n; x[3] += x1n; x[4] += x2n;
}"""

simulate_sde_src = """
// GPU kernel function to simulate trajectories of SDE
__kernel void simulate_sde(__global double *p, __global int *q,
     __global double *s)
{
// Indexing the current element (trajectory) to process
int i = get_global_id(0);
// Pointer to the i'th row of s (output)
__global double *s_i = &s[i*6]; // 6 values in output
// Simultaneous calculation of SDE trajectories within OpenCL kernel
double x[6]; // placeholders for s1,s2,x0,x1,x2,omega
```

```
int q_i = (int) q[i], *seed = &q_i;
const double r4_pi = 3.141592653589793;
double v1,v2;

for (int j = 0; j < 5; j++) { x[j] = p[P_S10+j]; }

for (int j = 0; j < (int) p[P_N]/2.; j++) { // 2 steps of SDE
   v1 = r8_uniform_01 ( seed );
   v2 = r8_uniform_01 ( seed );
   sde_step( p, x, sqrt(-2.0*log(v1))*cos(2.0*r4_pi*v2) );
   sde_step( p, x, sqrt(-2.0*log(v1))*sin(2.0*r4_pi*v2) );
}
for (int j = 0; j < 6; j++) { s_i[j] = x[j]; }
}"""
```

4. Compile the GPU C program from source:

```
opencl_program = cl.Program(opencl_context,
     sde_hdr_src+sde_rng_src+sde_step_src+simulate_sde_src).build()
```

5. Enqueue the SDE simulation program on the GPU device and wait until the program completion:

```
event = opencl_program.simulate_sde(command_queue, s_gpu.shape, None,
     p_buffer, q_buffer, s_buffer)
event.wait()
```

6. Get back the output data from GPU memory into NumPy array `s_gpu`:

```
cl.enqueue_copy(command_queue, s_gpu, s_buffer).wait()
```

Function `r8_uniform_01` is an implementation of uniform random number generator [26], C++ versions of other random number generators may be found in [27]. Standard normally distributed random variables $Z_{i,k}$ are implemented with Box–Muller transform [28].

Simulation of discretized equations (22) is performed using computer program in Python with the following parameters:

$$s_{1,0} = 0.9,\ s_{2,0} = 1.1,\ \mu_1 = 0.3,\ \sigma_1 = 0.4,\ \mu_2 = -0.1,\ \sigma_2 = 0.3,\ t_0 = 0.,\ T = 1.$$
$$x_{0,0} = 90.,\ x_{1,0} = 110.,\ x_{2,0} = 100.,\ r = 0.05,\ \alpha = 1.,\ \psi \equiv 0,\ N = 100.$$

The results of the computer experiment are shown in Fig. 1. Green line on the last plot with stock weights shows that the total share of stocks $w_1 + w_2$ tends to the value of 0.8, which is the target property of the portfolio under consideration.

Thus, constructed equations (22) generate trajectories with empirical portfolio property close to the given property (17).

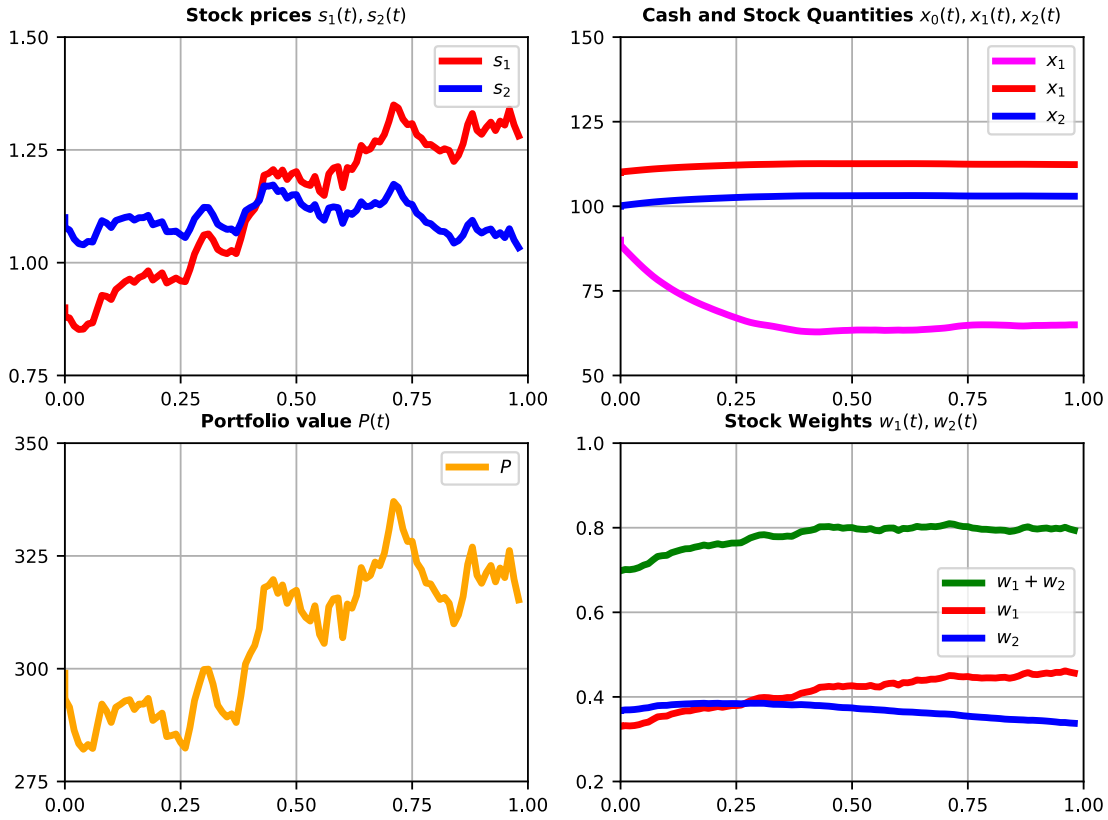

Figure 1: Simulation of portfolio with cash and two stocks and given structural ratio $w_1 + w_2 = 0.8$

Acknowledgments

The research was funded by RFBR, grant No. 19-08-00261.

References

[1] E. Allen, Modeling with Itô Stochastic Differential Equations, Mathematical Modelling: Theory and Applications, Springer Netherlands, 2007. doi:10.1007/978-1-4020-5953-7.

[2] N. V. Kampen, Stochastic differential equations, Physics Reports 24 (1976) 171–228. doi:10.1016/0370-1573(76)90029-6.

[3] R. P. King, Applications of stochastic differential equations to chemical-engineering problems: An introductory review, Chemical Engineering Communications 1 (1974) 221–237. doi:10.1080/00986447408960433.

[4] C. A. Braumann, Introduction to Stochastic Differential Equations with Applications to Modelling in Biology and Finance, Wiley, 2019. doi:10.1002/9781119166092.

[5] S. Primak, V. Kontorovich, V. Lyandres, Stochastic Methods and Their Applications to Communications, Wiley, 2004. doi:10.1002/0470021187.

[6] R. Bellman, A markovian decision process, Journal of Mathematics and Mechanics 6 (1957) 679–684. doi:10.1512/iumj.1957.6.56038.

[7] X. Y. Zhou, G. G. Yin, Markowitz's mean-variance portfolio selection with regime switching: A continuous-time model, SIAM J. Control. Optim. 42 (2003) 1466–1482. doi:10.1137/S0363012902405583.

[8] D. A. Pachamanova, F. J. Fabozzi, Portfolio Construction and Analytics, John Wiley & Sons, Inc, 2016. doi:10.1002/9781118656747.

[9] N. Erugin, Construction to all set of differential systems having a given invariant curve, Prikladnaia matematica i mehanika 16 (1952) 659–670.

[10] A. S. Galiullin, Certain problems in the design of programmed-motion systems, in: Symposia on Theoretical Physics and Mathematics 8, Springer US, 1968, pp. 185–192. doi:10.1007/978-1-4684-7721-4_16.

[11] R. G. Mukharlyamov, On the construction of differential equations of motion of constrained mechanical systems, Differential Equations 39 (2003) 369–380. doi:10.1023/a:1026021701825.

[12] R. G. Mukharlyamov, M. I. Tleubergenov, Control of system dynamics and constraints stabilization, in: Communications in Computer and Information Science, Springer International Publishing, 2017, pp. 431–442. doi:10.1007/978-3-319-66836-9_36.

[13] M. I. Tleubergenov, An inverse problem for stochastic differential systems, Differential Equations 37 (2001) 751–753. doi:10.1023/a:1019285119532.

[14] M. I. Tleubergenov, On the inverse stochastic reconstruction problem, Differential Equations 50 (2014) 274–278. doi:10.1134/s0012266114020165.

[15] M. Tleubergenov, and G.T. Ibraeva and, On inverse problem of closure of differential systems with degenerate diffusion, Eurasian Mathematical Journal 10 (2019) 93–102. doi:10.32523/2077-9879-2019-10-2-93-102.

[16] O. C. Ibe, Markov Processes for Stochastic Modeling, 2nd edition ed., Elsevier, 2013. doi:10.1016/c2012-0-06106-6.

[17] T. Björk, Arbitrage Theory in Continuous Time, 4th edition ed., Oxford University Press, 2019. doi:10.1093/oso/9780198851615.001.0001.

[18] G. Wang, Y. Wei, S. Qiao, Generalized Inverses: Theory and Computations, Springer Singapore, 2018. doi:10.1007/978-981-13-0146-9.

[19] S. Shorokhov, Management of Financial Asset Portfolios, RUDN University, 2013.

[20] G. N. Mil'shtejn, Approximate integration of stochastic differential equations, Theory of Probability & Its Applications 19 (1975) 557–562. doi:10.1137/1119062.

[21] P. E. Kloeden, E. Platen, Numerical Solution of Stochastic Differential Equations, Springer Berlin Heidelberg, 1992. doi:10.1007/978-3-662-12616-5.

[22] M. N. Gevorkyan, T. R. Velieva, A. V. Korolkova, D. S. Kulyabov, L. A. Sevastyanov, Stochastic Runge–Kutta software package for stochastic differential equations, in: Dependability Engineering and Complex Systems, Springer International Publishing, 2016, pp. 169–179. doi:10.1007/978-3-319-39639-2_15.

[23] G. Maruyama, Continuous markov processes and stochastic equations, Rendiconti del Circolo Matematico di Palermo 4 (1955) 48–90. doi:10.1007/bf02846028.

[24] A. Klöckner, N. Pinto, Y. Lee, B. Catanzaro, P. Ivanov, A. Fasih, PyCUDA and PyOpenCL: A scripting-based approach to GPU run-time code generation, Parallel Computing 38 (2012)

157–174. URL: https://doi.org/10.1016%2Fj.parco.2011.09.001. doi:10.1016/j.parco.2011.09.001.

[25] B. Gaster, L. Howes, D. R. Kaeli, P. Mistry, D. Schaa, Heterogeneous Computing with OpenCL, Elsevier, 2012. doi:10.1016/c2011-0-69669-3.

[26] J. Burkardt, UNIFORM – a uniform random number generator (RNG), 2019. URL: https://people.sc.fsu.edu/~jburkardt/c_src/uniform/uniform.html.

[27] M. N. Gevorkyan, A. V. Demidova, A. V. Korolkova, D. S. Kulyabov, L. A. Sevastianov, I. M. Gostev, Pseudo-random number generator based on neural network, in: CEUR Workshop Proceedings, volume 2267, 2018, pp. 568–572.

[28] G. E. P. Box, M. E. Muller, A note on the generation of random normal deviates, The Annals of Mathematical Statistics 29 (1958) 610–611. doi:10.1214/aoms/1177706645.