Aleksandar S. Cvetković
Full Professor
University of Belgrade
Faculty of Mechanical Engineering
Kraljice Marije 16, 11120 Belgrade 35

Nataša R. Trišović
Associate Professor
University of Belgrade
Faculty of Mechanical Engineering
Kraljice Marije 16, 11120 Belgrade 35

Wei Li
Associate Professor
Xidian University, No. 2, Taibai South Road,Xi'an, Shaanxi, 710071, Box 245
China

Simulation of The Linear Mechanical Oscillator on GPU

In this short paper we are describing some methods for solving stochastic differential equations which appears in mechanics. We are concerned with solutions of mechanical oscillators which are subject to random excitation. We are primarly interested in solving stochastic differential equations excited with white noise. We rewrite the random mechanical oscillator equation in the form suitable for applying numerical procedures for solving it. Finally, we give some design notes on the methods used to implement numerical simulation of the graphical processing units.

Keywords: *Graphics Processing Unit, Linear Mechanical Oscillator*

1. INTRODUCTION

Mathematical models in the form of Stochastic differential equations (SDE) with oscillating solutions can be found in various scientific areas: mechanical system [1], vibratory impact systems [2], statistical radio engineering [3], vibro-absorber systems [4], self-oscillating systems [5], oscillations of elasto-plastic bodies [6], statistical radio physics [7], and financial mathematics [8]. In solving numerically the oscillating ordinary differential equations (ODEs), satisfactory accuracy of calculation by the Euler method is usually obtained when 32 integration steps are used per oscillation period (see [26]).

Previous experiments [9], have shown that when oscillating SDEs with such an integration step are solved numerically, there often occurs instability in the numerical solution, that is, a great increase in the amplitude and variance of oscillations, whereas for the exact SDE solution the variance either does not increase or increase slightly as one moves along the integration interval. Therefore, in comparison to ODEs, the integration step must be decreased by several orders of magnitude. In addition, when small volumes of simulated trajectories of SDE solutions are used, absolutely incorrect estimate of solution moments may be obtained if the distribution densities are highly asymmetric.

With extremely small integration steps of the generalized Euler method and great volumes of simulated trajectories of SDE solutions, supercomputers with a large number of processors are needed to obtain satisfactory accuracy of numerical analysis in a reasonable calculation time. It should be noted that the use of some time-consuming methods that are more accurate than the Euler method, for instance, the Milstein method [10], does not speed up solving the problem, but, on the contrary, makes it more difficult. These methods and their error estimates have been received considerable attentions in the literatures, see [18], [19], and [20]. It should be noted that the use of some time-consuming methods that are more accurate than the Euler method, for instance, the Milstein method [10], does not speed up solving the problem, but, on the contrary, makes it more difficult. Some preliminary investigations of the authors on solving SDEs with increasing variance on supercomputers are described in [21].

In this paper we study equation of linear mechanical oscillator with constant coefficients:

$$\ddot{x}+2\zeta_0\omega_0\dot{x}+\omega_0^2x=f(t)+g(t)\psi(t) \tag{1.1}$$

where ψ is white noise random process which in every time instance represents Gaussian random variable with zero mean and variance equals to one.

We are going to reinterpret the results already known, for the solution of the general form of the stochastic differential equation which is equivalent to the equation (1.1), for our special case. The general results about the solution of the linear stochastic differential equation can be found in [12], [13], and [14]. We are going to give the forms of these solutions in the case of equation (1.1) in the section 2. We use these analytic solutions to compare the results with the simulation. We present implementation, in section 3, of the Euler method and analytic solution on Tesla C2075, with computational capability 2.0, with graphics driver TCC, product of NVIDIA Corporation. Finally, we give comparison of the results in Section 5.

2. ANALYTIC SOLUTION

Equation (1.1) can be written in the following form

$$\dot{X}=\begin{bmatrix}0 & 1\\ -\omega_0^2 & -2\zeta_0\omega_0\end{bmatrix}Xdt+\begin{bmatrix}0\\ f(t)\end{bmatrix}dt+\begin{bmatrix}0\\ g(t)\end{bmatrix}dW(t) \tag{2.1}$$

where $X=(x,\dot{x})$ and W is the Wiener process.

In the special case of the constant coefficients we can solve equation explicitly. For the general form of the solution see [12], [13], and [14].

Theorem 2.1. *Explicit solution of the equation (1.1) is given by*

Received: May 2014, Accepted: November 2014
Correspondence to: Dr Aleksandar Cvetković
Faculty of Mechanical Engineering,
Kraljice Marije 16, 11120 Belgrade 35, Serbia
acvetkovic@mas.bg.ac.rs
doi:10.5937/fmet1404317C

$$X(t)=\frac{e^{-C_0t}}{S_0}\begin{bmatrix} C_0\sin S_0t+S_0\cos S_0t & \sin S_0t \\ \omega_0^2\sin S_0t & -C_0\sin S_0t+S_0\cos S_0t \end{bmatrix}X_0+$$
$$+\frac{1}{S_0}\int_0^t e^{-C_0(t-s)}\begin{bmatrix} \sin(S_0(t-s)) \\ -C_0\sin S_0(t-s)+S_0\cos S_0(t-s) \end{bmatrix}\cdot$$
$$\cdot\left(f(s)ds+g(s)dW(s)\right)$$

where $C_0=\omega_0\zeta_0$, $S_0=\omega_0\sqrt{1-\zeta_0^2}$ *and* $X_0=X(0)$ *is random variable.*

Proof. We use equation (2.1) for the proof and rewrite it in the matrix notation

$$dX=\left(AX+F(t)\right)dt+G(t)dW(t),$$
$$A=\begin{bmatrix} 0 & 1 \\ -\omega_0^2 & -2\zeta_0\omega_0 \end{bmatrix},\ F=\begin{bmatrix} 0 \\ f \end{bmatrix},\ G=\begin{bmatrix} 0 \\ g \end{bmatrix}$$

We know that in general case, (see [12] and [13]), linear stochastic differential equation

$$dX=\left(AX+F(t)\right)dt+G(t)dW(t)$$

has solution

$$X(t)=e^{At}X_0+\int_0^t e^{A(t-s)}\left(F(t)ds+G(s)dW(s)\right).\quad (2.3)$$

It is easy to prove that

$$A=VDV^{-1},\ V=\begin{bmatrix} -\dfrac{\overline{\Sigma_0}}{\omega_0} & -\dfrac{\Sigma_0}{\omega_0} \\ 1 & 1 \end{bmatrix},$$
$$D=-\omega_0\begin{bmatrix} \Sigma_0 & 0 \\ 0 & \overline{\Sigma_0} \end{bmatrix},\ V^{-1}=\frac{1}{2i\sqrt{1-\zeta_0^2}}\begin{bmatrix} \omega_0 & \Sigma_0 \\ -\omega_0 & -\overline{\Sigma_0} \end{bmatrix},$$

where $\Sigma_0=\zeta_0+i\sqrt{1-\zeta_0^2}$. Accordingly, we can compute

$$e^{At}=Ve^{Dt}V^{-1}=V\begin{bmatrix} e^{-\omega_0\Sigma_0t} & 0 \\ 0 & e^{-\omega_0\overline{\Sigma_0}t} \end{bmatrix}V^{-1}\quad (2.4)$$
$$=\frac{e^{-C_0t}}{S_0}\begin{bmatrix} C_0\sin S_0t+S_0\cos S_0t & \sin S_0t \\ \omega_0^2\sin S_0t & -C_0\sin S_0t+S_0\cos S_0t \end{bmatrix}.\quad (2.5)$$

Using this computation and equation (2.3), we can produce stated result. q.e.d

Following results from Evans, we are in the position to give the following theorem.

Theorem 2.2. *We have*

$$E\left(X(t)\right)=\frac{e^{-C_0t}}{S_0}B(t)E(X_0)+$$
$$+\frac{1}{S_0}\int_0^t e^{-C_0(t-s)}\begin{bmatrix} \sin S_0(t-s) \\ -C_0\sin S_0(t-s)+S_0\cos S_0(t-s) \end{bmatrix}f(s)ds$$

and

$$\Sigma\left(X(t)\right)=\frac{e^{-2C_0t}}{S_0^2}B(t)\Sigma\left(X_0\right)B(t)^T+\frac{1}{S_0^2}\int_0^t e^{-2C_0s}D(s)g^2(t-s)ds$$

where Σ *is covariance matrix, and where*

$$B(t)=\begin{bmatrix} C_0\sin S_0t+S_0\sin S_0t & \sin S_0t \\ \omega_0^2\sin S_0t & -C_0\sin S_0t+S_0\cos S_0t \end{bmatrix}$$

$$D(t)=\begin{bmatrix} \sin^2 S_0t & -C_0\sin^2 S_0t+S_0\sin S_0t\cos S_0t \\ -C_0\sin^2 S_0t+S_0\sin S_0t\cos S_0t & \left(-C_0\sin S_0t+S_0\cos S_0t\right)^2 \end{bmatrix}$$

Proof. It can be proven using equations given in Evans, and already calculated expressions for e^{At} given in (2.4). q.e.d.

3. NUMERICAL METHODS

In order to be able to implement simulator for the equation (2.2) we give the following theorem.

Theorem 3.1. *Let interval of interest of the values of the stochastic process X be given by* $\left[0,T\right]$ *and suppose we are given partition* $t_k, k=0,\dots,N$, $t_k<t_{k+1}$, $k=0,\dots,N-1$, *with* $t_0=0$ *and* $t_N=T$. *For* $t\in\left[0,T\right]$, $t_k<t<t_{k+1}$, *define*

$$X(t)=\frac{e^{-C_0t}}{S_0}\begin{bmatrix} C_0\sin S_0t+S_0\cos S_0t & \sin S_0t \\ \omega_0^2\sin S_0t & -C_0\sin S_0t+S_0\cos S_0t \end{bmatrix}X_0$$
$$+\frac{1}{S_0}\sum_{l=0}^k e^{-C_0(t-t_l)}\begin{bmatrix} \sin S_0(t-t_l) \\ -C_0\sin S_0(t-t_l)+S_0\cos S_0(t-t_l) \end{bmatrix}\cdot$$
$$\cdot\left(f(t_l)\Delta t_l+g(t_l)\sqrt{\Delta t_l}\,\psi_l\right),\quad (3.1)$$

where ψ_k *are Gaussian random variables with zero means and unit variance and* $\Delta t_l=t_{l+1}-t_l$, $l=0,\dots,N-1$.

Proof. Fix $t\in\left[0,T\right]$and create partition $t_k\in\left[0,T\right]$, $k=0,\dots,N$, $t_k<t_{k+1}$, $t_0=0$, $t_N=T$. Choose k such that $t_k<t\le t_{k+1}$. According to Ito definition, see [11], of the stochastic integral we can approximate

$$\int_0^t e^{-C_0(t-s)}\begin{bmatrix} \sin S_0(t-s) \\ -C_0\sin S_0(t-s)+S_0\cos S_0(t-s) \end{bmatrix}g(s)dW(s)$$

$$\approx\sum_{l=0}^k e^{-C_0(t-t_l)}\begin{bmatrix} \sin S_0(t-t_l) \\ -C_0\sin S_0(t-t_l)+S_0\cos S_0(t-t_l) \end{bmatrix}g(t_l)\left(W(t_{l+1})-W(t_l)\right)$$
$$=\sum_{l=0}^k e^{-C_0(t-t_l)}\begin{bmatrix} \sin S_0(t-t_l) \\ -C_0\sin S_0(t-t_l)+S_0\cos S_0(t-t_l) \end{bmatrix}g(t_l)\sqrt{\Delta t_l}\,\psi_l$$

where we have used the fact that increment of the Wiener process $W(t_{l+1})-W(t_l)$ is Gaussian random variable with zero mean and variance Δt_l.

According to definition of the Ito stochastic integral, we know that expression to the right is converging to the integral on the left. Hence, it follows

$$\int_0^t e^{-C_0(t-s)}\begin{bmatrix} \sin S_0(t-s) \\ -C_0\sin S_0(t-s)+S_0\cos S_0(t-s) \end{bmatrix}g(s)dW(s)$$
$$=\lim_{\rho\to 0}\sum_{l=0}^k e^{-C_0(t-t_l)}\begin{bmatrix} \sin S_0(t-t_l) \\ -C_0\sin S_0(t-t_l)+S_0\cos S_0(t-t_l) \end{bmatrix}\cdot$$

$\cdot g(t_l)\sqrt{\Delta t_l}\,\psi_l$

Similarly according to the Riemann definition of the integral and according to the fact that integral is smooth, we know that

$$\int_0^t e^{-C_0(t-s)}\begin{bmatrix}\sin S_0(t-s)\\ -C_0\sin S_0(t-s)+S_0\cos S_0(t-s)\end{bmatrix}f(s)ds$$

$$=\lim_{\rho\to 0}\sum_{l=0}^{k}e^{-C_0(t-t_l)}\begin{bmatrix}\sin S_0(t-t_l)\\ -C_0\sin S_0(t-t_l)+S_0\cos S_0(t-t_l)\end{bmatrix}f(t_l)\Delta t_l$$

Combining this two facts statement of the theorem follows. q.e.d.

In order to be able to implement an efficient simulator, we can rewrite equation (3.1) into the following form

$$X(t)=\frac{e^{-C_0t}}{S_0}\begin{bmatrix}C_0\sin S_0t+S_0\cos S_0t & \sin S_0t\\ \omega_0^2\sin S_0t & -C_0\sin S_0t+S_0\cos S_0t\end{bmatrix}X_0$$
$$+\frac{e^{-C_0t}}{S_0}\begin{bmatrix}C_k\sin S_0t-S_k\cos S_0t\\ (-C_0C_k+S_0S_k)\sin S_0t+(C_0S_k+S_0C_k)\cos S_0t\end{bmatrix},$$

where

$$C_k=\sum_{l=0}^{k}e^{C_0t_l}\cos S_0t_l\left(f(t_l)\sqrt{\Delta t_l}+g(t_l)\right)\sqrt{\Delta t_l},$$

$$S_k=\sum_{l=0}^{k}e^{C_0t_l}\sin S_0t_l\left(f(t_l)\sqrt{\Delta t_l}+g(t_l)\right)\sqrt{\Delta t_l}.$$

Using this notation we can express C_k and S_k recursively

$$C_{k+1}=C_k+e^{C_0t_k}\cos S_0t_k\left(f(t_k)\sqrt{\Delta t_k}+g(t_k)\right)\sqrt{\Delta t_k},$$
$$S_{k+1}=S_k+e^{C_0t_k}\cos S_0t_k\left(f(t_k)\sqrt{\Delta t_k}+g(t_k)\right)\sqrt{\Delta t_k},$$

which gives us opportunity to solve equation efficiently. Even more, we can use the following recurrence relation

$$\begin{bmatrix}\cos S_0t_{k+1}\\ \sin S_0t_{k+1}\end{bmatrix}=\begin{bmatrix}\sin S_0\Delta t_k & \cos S_0\Delta t_k\\ \cos S_0\Delta t_k & -\sin S_0\Delta t_k\end{bmatrix}\begin{bmatrix}\cos S_0t_k\\ \sin S_0t_k\end{bmatrix},$$

which is especially useful in the case of the constant step $\Delta t_k, k=0,...,N-1$, since it eliminates computation of the cos and sin function in every step of the loop.

The only problem that might occur is connected to the fact that for significant values of t_l, computation of C_k and S_k can become unstable. In order to be able to compute for large values of t multiplication with e^{-C_0t} should be included in computation of C_k and S_k. In this case we define

$$e^{-C_0t}C_{k+1}=e^{-C_0t}C_k+e^{-C_0(t-t_k)}\cos S_0t_k\left(f(t_k)\sqrt{\Delta t_k}+g(t_k)\right)\sqrt{\Delta t_k},$$
$$e^{-C_0t}S_{k+1}=e^{-C_0t}S_k+e^{-C_0(t-t_k)}\sin S_0t_k\left(f(t_k)\sqrt{\Delta t_k}+g(t_k)\right)\sqrt{\Delta t_k},$$

However, if we want to continue computation for some moment $t'>t$, we must renormalize C_k and S_k with the factor $e^{-C_0(t'-t)}$.

There is also another method which can be used for computation. Actually, it is numerical method which can be used for the computation of the solution of general stochastic differential equations.

Theorem 3.2. *Let interval of interest for the solution of* X *be* $[0,T]$, *and define partition of the interest by the points* t_k, $k=0,...,N$, $t_0=0$, $t_n=T$ *and* $\Delta_k=t_{k+1}-t_k$, $k=0,...,N-1$.

Define $\rho=\max_{l=0,...,N-1}\Delta t_l$ *and*

$$X_{k+1}=X_k+\begin{bmatrix}0 & 1\\ -\omega_0^2 & -2\zeta_0\omega_0\end{bmatrix}X_k\Delta t_k$$
$$+\left(\sqrt{\Delta t_k}\begin{bmatrix}0\\ f(t_k)\end{bmatrix}+\psi_k\begin{bmatrix}0\\ g(t_k)\end{bmatrix}\right)\sqrt{\Delta t_k},$$

for $k=0,...,N-1$.

There exist γ *such that for every* $k=1,...,N$, *there exists* C_k *and* $E\left|X(t_k)-X_k\right|\le C_k\rho^\gamma$.

It can be proved that $\gamma=1$.

Proof. This is well-known Euler-Maruyama method and bound for error can be found in Higham (2001). However, this is result for the Euler-Maruyama method in the case of the general form of the stochastic differential equation, and our equation is the special form in which case Euler-Maruyama method is equivalent to the Milstein method and Mistein method has $\gamma = 1$ (see [15]). q.e.d.

Actually, it can be shown that constant C_k in the previous theorem depends exponentially on t_k, which means that stability of computation is not guaranteed for the presented method. Actually, this exponential behavior requires that ρ should be small in order to be able to compute with requested precision, otherwise computation break down rapidly.

4. IMPLEMENTATION DETAILS

In this section we give implementation details for the methods presented in theorems 3.1 and 3.2. We are interested in the computation of the probabilty density function for the random variable $X(t)$, for some fixed moment t > 0. According to the architecture of the graphic processing unit, the obvious parallelization is to compute one realization of $X(t)$ in every thread. This is rather obvious since we are working on the SIMT architecture and all threads in the simulation are going to compute the same instruction over different data sets, which guaranties optimal performance (see [17], [22], [23], [24], [25]).

In order to implement simulations on parallel architecture, we use the fact that every sample path is independent of any other sample path. Hence, we implement in every thread one sample path simulation. We are going to present implementation details for the NVIDIA graphic processing unit Tesla C2075, with compute capability 2.0 and driver model TCC.

We use simulation on the interval [0,T] with number of points *N*, which are equidistantly distrubuted by $t_i = ih,\ h = T/N,\ i = 0,...,N$. Parameters that are needed for simulation are ζ_0, ω_0, and functions *f* and *g*. Functions *f* and *g* are implemented as device functions and are specified for the concrete simulation. Parameters ζ_0, ω_0 are transferred to the device in the constant memory. We use constant memory to transfer parameters ζ_0, ω_0, step size $h, \sqrt{h}$ and initial displacement $(X_1)_0$ and velocity $(X_2)_0$ to the graphical processing unit. This means that we have seven variables of type double in constant memory. These are not the only variables we need to pass to the constant memory. Obviously we cannot store results of the computation in every time step, since, due to the limitation of the 4GB of total memory. Rather, we pass indexes of time steps in which we want to store results. Indexes of this time steps are passed to the graphical processing unit using integer values in the constant memory.

We launch kernel using parameters <<< numOfBlocks, numOfThread >>>. Parameter numOfBlocks amounts to the total number of Blocks kernel is launched for. Architecture of the graphical processing unit we are using shows that this number can not be bigger than 2^{16} = 65536. Parameter numOfThread, however, has maximal possible value 2^{10} = 1024. The problem with achieving this value for the amount of possible number of threads is rather hard to achieve since there is the problem with maximal number of registers which can be assigned to the every thread. Namely, maximal number of registers per block for this architecture is limited to the value of 2^{15} = 32768. However, every thread needs to compute random variable which has Gaussian distribution and every thread need to store variables needed for the computation. Those variables are ζ_0, ω_0, step size $h, \sqrt{h}$, and variables x_1 and x_2 which represent current value of the computation of X_1 and X_2. There are also variables needed to compute functions *f* and *g* which additionally occupy total number of registers. In total, using careful implementation we can achieve launch of 512 threads per block. Random variables with Gaussian distribution are computed using library function curand_double_normal. Additionally, for the method presented in the theorem 3.1 we use CUDA library functions cos, sin and exp. Extensive usage of this function however, increases complexity of the computation on the CUDA device with respect to the method presented in the theorem 3.2.

After computation of the solutions of the stochastic differential equation, we need to compute probability density functions for the random variables X_1 and X_2. Computation can be achieved using the following formula

$$F_{X_i}(a) = P(X_i < a) = \frac{\text{number of realizations in favor}}{\text{total number of realizations}}, \quad i = 1,2,$$

where total number of realizations is simply product of numOfBlocks and numOfThread and *number of realizations in favor* is the number of realizations of X_i, $i = 1,2$, which satisfy property $X_i < a$, $i = 1,2$.

After simulation is completed we are in the position to compute probability density function of the variables X_i, $i = 1,2$. Firstly, we compute distribution function. However, in order to be able to compute distribution function we first sort realization of the variables X_i, $i = 1,2$. We sort using function sort of the thrust library. Sorting can be achieved with at most ten present of the total computational time. After sorting the ensembles we divide realizations of the variables X_i, $i = 1,2$, in the small pieces and compute distribution function on the small segments of realizations. Distribution function on these small segments of the samples can be achieved using one thread per segment. Using this implementation for the computation can be achieved using at most one present of time needed for the computation of the realizations. However, since we do not return data to the central processing unit memory, we need to pass parameters for this computation of the distribution function into the constant memory. This amounts in additional two integers passed to the graphical processing unit using constant memory.

5. NUMERICAL RESULTS

We have used method presented in the previous section and implemented it on Tesla C2075, with compute capability 2.0 and driver model TCC. In this section we present some numerical results. We present results for the linear oscillator with parameters $\zeta_0 = 0.02$, $\omega_0 = 1$, $f = 0$ and $g = 1$. We assume that ψ is the Gaussian process with flat power spectral density *S* = 0.1. We present results for 512 numbers of threads and 2000 blocks. In total this means that we are using 1024000 realizations of the Wiener process. We have chosen to compute probability density function for the time instances 20, 40, 60, 80 and 100.

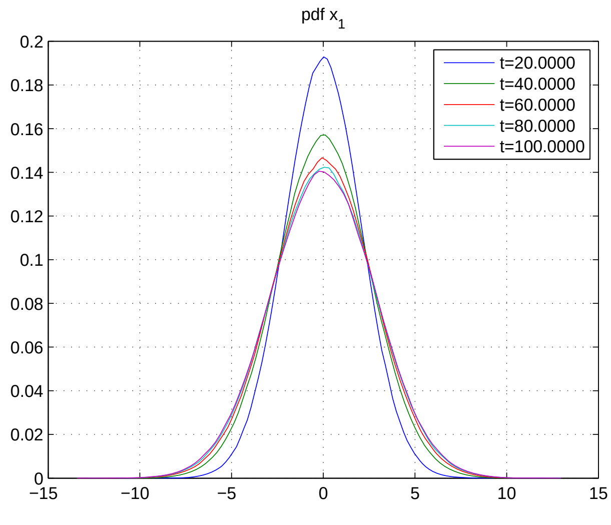

Figure 1. Probability density function for the displacement for the linear simulator with $\zeta_0 = 0.02$, $\omega_0 = 1$, $f = 0$ and $g = 1$, with the power spectral of the Gaussian noise $S = 0.1$

Figure 1 holds results for the probability density function for the displacement, while figure 2 holds results for the probability density function for velocity. Also figure 3 holds results for the envelope. Envelope is computed using formula $X_e^2 = X_1^2 + X_2^2 / \omega_0^2$.

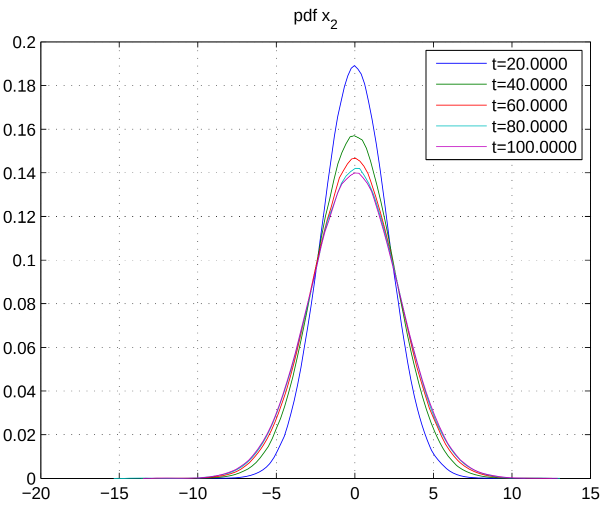

Figure 2. Probability density function for the velocity for the linear simulator with $\zeta_0 = 0.02,\ \omega_0 = 1,\ f = 0$ and $g = 1$, with the power spectral of the Gaussian noise $S = 0.1$

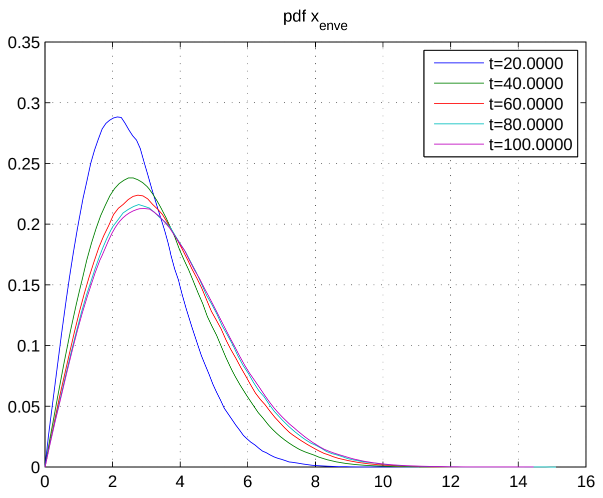

Figure 3. Probability density function for the envelope for the linear simulator with $\zeta_0 = 0.02,\ \omega_0 = 1,\ f = 0$ and $g = 1$, with the power spectral of the Gaussian noise $S = 0.1$

6. CONCLUSION

In this short paper we studied possibility of simulating linear oscillator with random excitation on graphical processing units. We give description of the implementation we used and report that we are in position to run over million simulations in parallel. Complexity of running lot of simulations is spread over the significant number of threads on the graphical processing unit which can speed up computation by the factor of one hundred.

ACKNOWLEDGMENT

This research was performed within the "Chinese-Serbian Science & Technology Cooperation" (No. 2-14) projects supported by Ministry of Science and Technological Development, Republic of Serbia, whose funding is gratefully acknowledged.

REFERENCES

[1] Dimentberg, M. F.: Nelineinnye stokhasticheskie zadachi mekhanicheskikh kolebanii (Nonlinear Stochastics Problems of Mechanical Oscillations), Moscow: Nauka, 1980.

[2] Babitskii, V.I.: Teoriya vibroudarnykh system (Theory of Vibro-Impact Systems), Moscow: Nauka, 1978.

[3] Bykov, V.V.: Tsifrovoe modelirovanie v statisticheskoi radiotekhnike (Digital Modeling in Statistical Radio Engineering), Moscow: Sovetskoe radio, 1971.

[4] Kolovskii, M.Z., Nelineinaya teoriya vibro zashchitnykh system (Nonlinear Theory of Vibro-Absorber Systems), Moscow: Nauka, 1966.

[5] Malakhov, A.N.: Fluktuatsii v avtokolebatel'nykh sistemakh (Fluctuations in Self-Oscillating Systems), Moscow: Nauka, 1968.

[6] Pal'mov, V.A.: Kolebaniya uprugo-plasticheskikh tel (Oscillations of Elasto-Plastic Bodies), Moscow: Nauka, 1976.

[7] Rytov, S.M.: Vvedenie v statisticheskuyu radio fiziku (Introduction to Statistical Radio Physics), Moscow: Nauka, 1966.

[8] Artemiev, S.S. and Yakunin, M.A.: Matematicheskoe i statisticheskoe modelirovanie v finansakh (Mathematical and Statistical Modeling in Finances), Novosibirsk: Inst. Comp. Math. Math. Geophys., SB RAS, 2008.

[9] Artemiev, S.S.: Chislennye metody resheniya zadachi Koshi dlya sistem obyknovennykh i stokhasticheskikh differentsialnykh uravnenii (Numerical Methods of Solving Cauchy Problems for Systems of Ordinary and Stochastic Differential Equations), Novosibirsk: Inst. Comp. Math. Math. Geophys., SB RAS, 1993.

[10] Milstein, G.N.: Chislennoe integrirovanie stokhasticheskikh diff erentsialnykh uravnenii (Numerical Integration of Stochastic Differential Equations), Sverdlovsk: SSU, 1988.

[11] Roberts, J.B., Spanos, P.D.: Random Vibrations and Statistical Linearization, Dover Publications, New York, 1999.

[12] L.C. Evans: Introduction to Stochastic Differential Equations, Berkley lecture notes, 2002.

[13] Arnold, L.: Stochastic Differential Equations: Theory and Practise, John Wiley and Sons, New York, 1974.

[14] Lyn, Z.K.: Probabilistic Theory of Structural Dynamics, McGRAW-HILL, 1967.

[15] Higham, D.J.: An Algorithmic Introduction to Numerical Simulation of Stochastic Differential Equations, SIAM Review, 43(3), pp. 525-546, 2001.

[16] Allen, A.: Modelling with Ito Stochastic Differential Equations, Springer, 2007.

[17] Farber, R.: CUDA Application Design and Development, Morgan Kaufmann, USA, 2011.

[18] Kloeden, P. E. and Platen, E.: Numerical solution of stochastic differential equations, Springer, 1999.

[19] Oksendal, B. K.: Stochastic differential equations: an introduction with applications, Springer, 2000.

[20] Henderson, D. and Plaschko, P.: Stochastic differential equations in science and engineering. World Scientific Publishing, 2006.

[21] Artemiev, S.S. and Korneev, V.D.: Numerical Solution of Stochastic Differential Equations on Supercomputers, Sib. Zh. Vych. Mat., vol. 14, no. 1, pp. 5, 2011.

[22] Korneev, V.D.: Parallel'noe programmirovanie v MPI (Parallel Programming in MPI), Moscow Izhevsk: Inst. Comp. Sci., 2003.

[23] Korneyev, V.D.: Parallel'noe programmirovanie klasterov. Uchebnoe posobie (Parallel Programming of Clusters: Tutorial), Novosibirsk: N ovosibirsk State Techn. Univ., 2008.

[24] http://www.nvidia.com/object/cuda_home_new.htm l

[25] http://mc.stanford.edu/cgi-bin/images/b/ba/M02_2.pdf

[26] Artemiev, S.S., Ivanov, A.A., Korneev: Numerical Analysis of Stochastic Oscillators on Supercomputers, Numerical Analysis and Applications, Vol. 5, No. 1, pp. 25-35, 2012.

СИМУЛАЦИЈА ЛИНЕАРНИХ МЕХАНИЧКИХ ОСЦИЛАТОРА НА ГПУ

Александар Цветковић, Наташа Тришовић, Wei Li

У овом кратком раду ми смо описали неке методе за решавање стохастичких диференцијалних једначина које се јављају у механици. Пре свега нас занимају механичке осцилације које настају случајном побудом. Случајна побуда у облику белог шума се најчешће користи за теоријска и практична разматрања, зато и ми бирамо побуду у облику белог шума. Стандардни облик једначине механичког осцилатора је записан у облику који је погодан за нумеричко решавање. На крају дајемо неке елементе писања програма на ГПУ.