Ministry of Education and Science of Ukraine
Sumy State University

4342 METHODOLOGICAL INSTRUCTIONS

for practical training
on the discipline “Nonlinear processes and models”
on the topic “Modelling of random process”
for students of specialty 113 “Applied mathematics”
full-time training

Sumy
Sumy State University
2018

Methodological instructions for practical training on the discipline “Nonlinear processes and models” on the topic “Modelling of random process” / Compilers O. V. Khomenko, A. M. Zaskoka. – Sumy : Sumy State University, 2018. – 35 p.

Department of Applied Mathematics and Complex System Modelling

CONTENT

INTRODUCTION

Melting of an ultrathin lubricant film during friction between atomically smooth surfaces is studied. Additive noise of shear stress and strain as well as of film temperature is introduced and the phase diagram is constructed. On the diagram, the noise intensity for this temperature and the temperature of friction surfaces determine the regions of sliding, dry, and stick-slip friction. As a result of numerical analysis of the Langevin equation for various regions of the diagram, time series of stresses are constructed, which make it possible to explain the experiment on friction, in which intermittent motion is observed. Lubricant melting due to dissipative heating of friction surface is considered and the experimental time dependences of friction force are interpreted.

1 BASIC EGUATIONS

Melting of an ultrathin lubricant film in the course of friction between atomically smooth solid surfaces with allowance for additive noise of shear stress and strain as well as of temperature is described by the Langevin equation [1, 2]

$$\tau_\sigma \dot{\sigma} = f(\sigma) + \sqrt{I(\sigma)}\xi(t), \tag{1}$$

where σ is the shear stress (which is an order parameter) and τ_σ is its relaxation time. For $\sigma = 0$, the lubricant is solidlike, while the case $\sigma \neq 0$ corresponds to its liquidlike state [3]. In accordance with the new friction map for the boundary regime [4], an increase in stress

$$\sigma = \frac{F}{A}, \tag{2}$$

leads to an increase in the viscous friction force

$$F = \frac{\eta_{eff} VA}{h}, \tag{3}$$

where V is the velocity of a block, h is the film thickness, η_{eff} is the effective viscosity, and A is the contact area. Comparing relations (2) and (3), we obtain the following expression for velocity in terms of stress:

$$V = \sigma \frac{h}{\eta_{eff}}, \tag{4}$$

in other words, the total friction force decreases upon an increase in stresses since the velocity of the moving surfaces increases (the lubricant melts).

Generalized force $f(\sigma)$ and effective noise intensity $I(\sigma)$ are given by the equalities

$$\begin{aligned} f(\sigma) &\equiv -\sigma + g\sigma\left[1-(2-T_e)d(\sigma)\right], \\ I(\sigma) &\equiv I_\sigma + \left(I_\varepsilon + I_T\sigma^2\right) g^2 d^2(\sigma), \\ d(\sigma) &\equiv \left(1+\sigma^2\right)^{-1}. \end{aligned} \tag{5}$$

Here, T_e is the temperature of friction surfaces; $g(\sigma) = G/G_0 \equiv \text{const}$ is the ratio of the shear modulus to its characteristic value; and I_σ, I_ε, and I_T are the intensities of stress, strain, and temperature noise, respectively. Function $\xi(t)$ in Eq. (1) is a δ-correlated stochastic source (white noise). Its moments are defined as

$$\left\langle \xi_i(t) \right\rangle = 0, \quad \left\langle \xi_i(t)\xi_j(t') \right\rangle = 2D\delta_{ij}\delta(t-t'), \tag{6}$$

where D is the integral of the correlation function, which plays the role of the source intensity. The distribution of the values of $\xi(t)$ over ξ is Gaussian [5]:

$$P(\xi) = \frac{1}{\sqrt{2\pi}\mu} \exp\left\{-\frac{\xi^2}{2\mu^2}\right\}. \quad (7)$$

Here, μ^2 is the second moment of the source,

$$\mu^2 \equiv \left\langle \xi^2(t) \right\rangle = 2D\delta(0), \quad (8)$$

which diverges as D/τ, where $\tau \to 0$ is the width of the δ function, which assumes a nonzero value for all real physical systems.

Langevin equation (1) is in one-to-one correspondence with the Fokker–Planck equation (FPE) in the Ito form [6]

$$\tau_\sigma \frac{\partial P(\sigma,t)}{\partial t} = -\frac{\partial}{\partial \sigma} f(\sigma) P(\sigma,t) + D \frac{\partial^2}{\partial \sigma^2} \left(I(\sigma) P(\sigma,t) \right). \quad (9)$$

The distribution of solutions to Eq. (1) becomes stationary after some time and its explicit form can be found from Eq. (9) for $\partial P(\sigma,t)/\partial t = 0$:

$$P(\sigma) = Z^{-1} \exp\left\{-U(\sigma)\right\}. \quad (10)$$

The resultant distribution is controlled by the normalization constant

$$Z = \int_{-\infty}^{+\infty} d\sigma \exp\left\{-U(\sigma)\right\} \quad (11)$$

and the effective potential[1]

$$U(\sigma) = \ln I(\sigma) - \frac{1}{D} \int_{-\infty}^{\sigma} \frac{f(\sigma')}{I(\sigma')} d\sigma'. \quad (12)$$

The equation defining the positions of maxima of function $P(\sigma)$ has the form

[1] In contrast to [1], the lower integration limit in our case is $-\infty$ and not 0 since negative stresses will also be considered below.

$$(1-g)x^3+g(2-T_e)x^2-2gDI_Tx+4g^2D(I_T-I_\varepsilon)=0, \qquad x\equiv 1+\sigma^2. \tag{13}$$

Thus, the shape of the $P(\sigma)$ distribution is independent of noise intensity I_σ.

2 PROCEDURE OF CONSTRUCTION OF PHASE DIAGRAMM AND TIME SERIES

For a fixed intensity I_ε, the phase diagram has the form shown in Fig. 1, where curves I and II correspond to stability loss limits for the system. The line II is built by numerical searching of extrema of Eq. (13) (see Append. A). Above straight line I defined by the equation

$$T^c=1+g^{-1}+2gD(I_T-2I_\varepsilon), \tag{14}$$

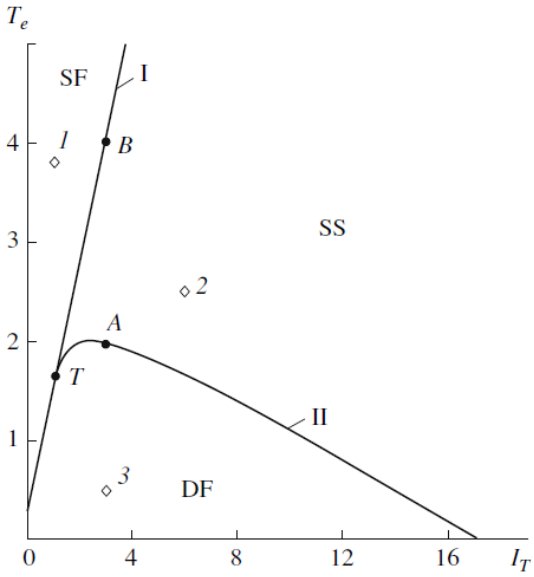


Fig. 1. Phase diagram for g = 0.8, I_ε = 0.8, and D = 0.8 with the regions of sliding (SF), dry (DF), and stick-slip (SS) friction (T is the tricritical point) (look Append. A).

the most probable value of stress $\sigma \neq 0$ and the lubricant is in the liquidlike phase ensuring stable sliding friction (SF) state. Below curve II, which is tangential to straight line I at tricritical point T with coordinates

$$T_e^c = \frac{2}{3}\left(1+2g^{-1}-2gDI_\varepsilon\right), \quad I_T^c = \frac{1}{6gD}\left(g^{-1}-1+8gDI_\varepsilon\right), \quad (15)$$

function $P(\sigma)$ has a peak only at point $\sigma = 0$ and dry friction (DF) typical of solidlike lubricant takes place. Between these curves, the peaks of $P(\sigma)$ correspond to zero and nonzero value of stresses; i.e., this is the region of stick-slip (SS) friction, where random transitions between these dynamic friction regimes occur. This is typical of the intermittent conditions during lubricant melting, where a mixture of solidlike and liquidlike states exists. In accordance with expressions (15), lubricant can melt even at zero temperature T_e of the friction surfaces if the intensity of strain fluctuations exceeds the critical value

$$I_\varepsilon = \left(1+2g^{-1}\right)\Big/2gD.$$

It follows hence that this value decreases upon an increase in the stochastic source intensity D.

The phase diagram of states of the system is constructed in accordance with the analytically determined distribution $P(\sigma)$. We will carry out such a procedure analyzing Eq. (1) numerically. This is due to the fact that time dependences of physical quantities are usually measured in experiments. In addition, this procedure makes it possible to demonstrate once again that the FPE was chosen correctly.

Multiplying Eq. (1) by dt, we obtain the Langevin differential equation

$$\tau_\sigma d\sigma = f\left(\sigma\right)dt + \sqrt{I\left(\sigma\right)}dW\left(t\right), \quad (16)$$

where $dW(t) = W(t + dt) - W(t) \equiv \xi(t)dt$ is a Wiener process exhibiting the properties of white noise [7]:

$$\left\langle W(t)\right\rangle = 0, \quad \left\langle \left(W(t)\right)^2 \right\rangle = 2Ddt. \tag{17}$$

To solve Eq. (16) numerically, we will employ the Euler method. Having measured time in the τ_σ units and taking into account the definition of a discrete analog of random force differential $dW(t) \equiv \sqrt{\Delta t}\, W_n$, we obtain the iterative procedure for calculating the time series of stresses:

$$\sigma_{n+1} = \sigma_n + f(\sigma_n)\Delta t + \sqrt{I(\sigma_n)\Delta t} W_n. \tag{18}$$

The equation is solved on time interval $t \in [0, T]$. For a preset number N of iterations (number of points of the time series), the time increment is defined as $\Delta t = T/N$. Force W_n possesses the following properties:

$$\left\langle W_n \right\rangle = 0, \quad \left\langle W_n W_{n'} \right\rangle = 0, \quad \left\langle W_n^2 \right\rangle \to 2D. \tag{19}$$

A random force possessing the properties of white noise can be correctly defined using the Box–Muller model [8]:

$$W_n = \mu\sqrt{-2\ln r_1}\cos(2\pi r_2), \quad r_i \in (0,1], \tag{20}$$

where $\mu = \sqrt{2D}$ and W_n is an absolutely random number with properties (19) and (7). Pseudorandom numbers r_1 and r_2 are repeated after a certain interval.

Trajectories $\sigma(t)$ for various regions on the phase diagram are shown in Fig. 2. The upper part of the figure corresponds to point *1* in Fig. 1 (SF). One can see casual transitions between positive and negative stable values of stress of the same absolute magnitude. This regime corresponds to the liquidlike structure of the lubricant and a small value of friction force. We must assume that reverse motion (motion of surfaces displaced in the opposite direction) takes place for negative values of stress; strain in this case is also negative. We can disregard the negative region, assuming that it is deprived of physical meaning, and suppose that

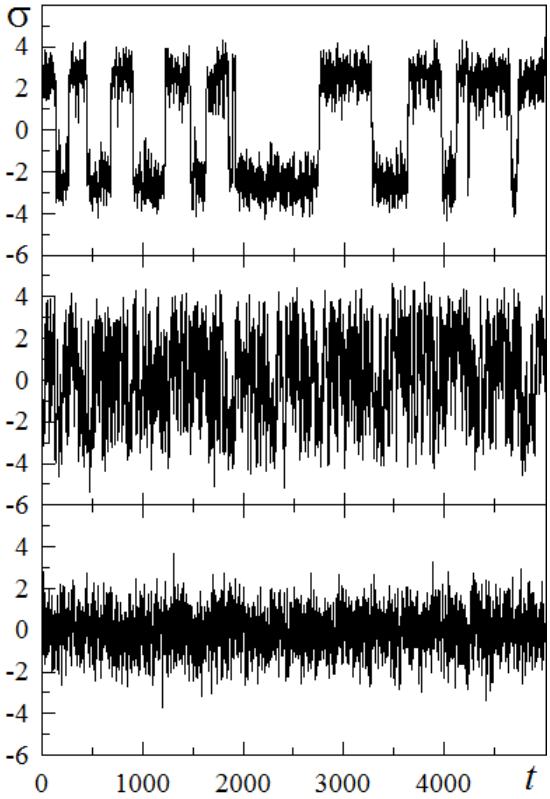


Fig. 2. Time series of stresses $\sigma(t)$ corresponding to the regimes represented by points in Fig. 1 for $I_\sigma = 0.1$. The upper part corresponds to point *1* ($I_T = 1$, $T_e = 3.8$, SF), the middle part corresponds to point *2* ($I_T = 6$, $T_e = 2.5$, SS), and the lower part corresponds to point *3* ($I_T = 3$, $T_e = 0.5$, DF) (look Append. B).

stresses begin to increase after attaining zero value, which allows us to analyze the behavior of $|\sigma|$. The middle part of the figure corresponds to point *2* in Fig. 1 (SS). It shows frequent transitions between zero and nonzero values of stress, which correspond to stick-slip friction, in which the friction force changes abruptly during motion. Since these transitions are very frequent, this situation corresponds to the intermittent conditions, when a mixture of liquidlike and solidlike phases exists in the lubricant

[2]. The lower part of the figure corresponds to point *3* in Fig. 1 (DF). In this case, oscillations occur in the vicinity of $\sigma = 0$, which corresponds to the solidlike structure of the lubricant and the highest value of the friction force.

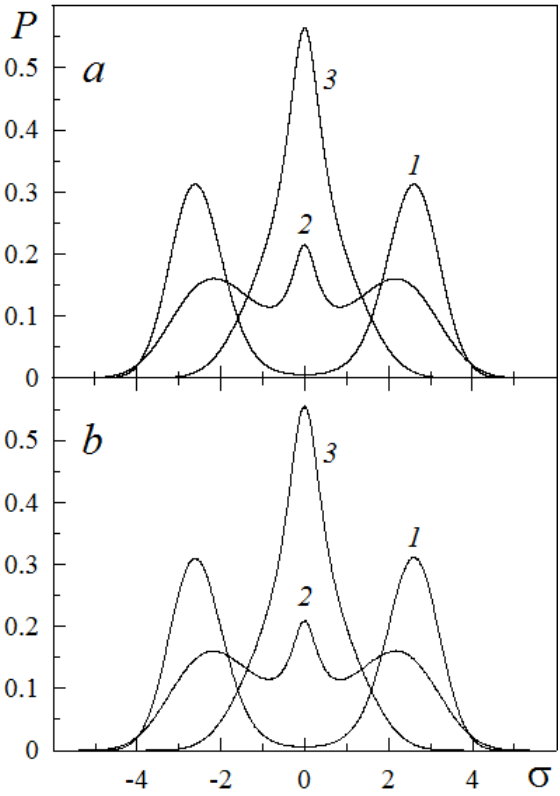


Fig. 3. Probability distributions: (a) determined by expression (10) and corresponding to points in Fig. 1; (b) determined from numerical analysis of Eq. (18) with parameters $N = 10^8$, $T = 10^6$, $\Delta t = 0.01$ and corresponding to trajectories in Fig. 2 (look Append. C).

Figure 3a shows probability distribution (10) for the points indicated on the phase diagram, while Fig. 3b shows analogous distributions obtained numerically by solving the Langevin equation and corresponding to the trajectories in Fig. 2. It can be seen that the corresponding dependences $P(\sigma)$ perfectly match in

both cases, which confirms the correspondence of the FPE in Ito form (9) to iterative procedure (18). Curve *1* corresponds to the region of sliding friction on the phase diagram, where only the nonzero maximum of $P(\sigma)$ exists; curve *2* corresponds to the stick-slip friction region, where zero and nonzero maxima exist; and curve *3* corresponds to the region of dry friction with a single zero maximum of $P(\sigma)$.

The experimental dependences of the friction force on the velocity of mica surfaces being displaced, between which hydrocarbon and silicon liquids (cyclohexane, octamethyl cyclotetrasiloxane, *n*-octane, *n*-tetradecane, and branched isoparaffin-2-methyl octadecane) were placed, are given in [9]. According to these curves, the friction force first increases linearly and then the stick-slip regime sets in, in which the friction force varies periodically, ensuring intermittent motion. In the approach proposed in [9], such a behavior can be explained as follows. At the first stage of motion, the elastic component of stress σ predominates and the lubricant is solidlike (the lower part of Fig. 2). In this region, Hooke's law is observed and the total friction force increases. For a certain critical value of velocity, transitions between the solidlike and liquidlike structures (in the latter structure, the viscous component of stress dominates) of the lubricant take place due to the "shear melting" effect [10] (middle part of Fig. 2).

In addition, the time dependences of the friction force, on which the melting of the lubricant is shown for constant shear rate are depicted (see, for example, [9, 11]), are measured experimentally. It should be noted that friction is always accompanied by dissipation of energy of translatory motion of rubbing surfaces, which leads to their heating. The dissipation is the stronger the higher the value of the total friction force; consequently, for the solidlike state of the lubricant, the surfaces are heated at a higher rate than in the case of the liquidlike state of the lubricant. The change in the temperature of the surfaces becomes weaker and weaker with time due to the increase in the

amount of energy liberated to the environment until the equilibrium value T_e sets in.

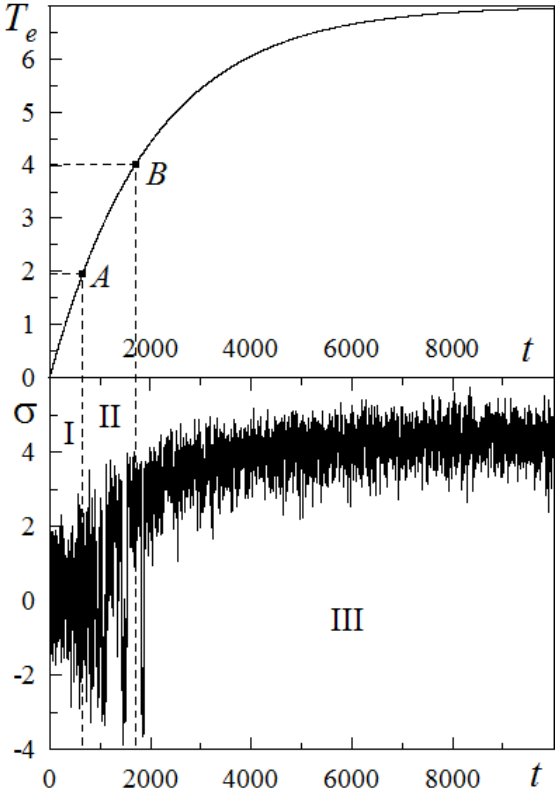


Fig. 4. Upper part shows the time dependence (21) of the temperature of moving surfaces for T_e^0 = 7.0, and C = 0.005. Points A and B correspond to points in Fig. 1. The lower part shows the time series $\sigma(t)$, corresponding to parameters in Fig. 1 for I_σ = 0, I_T = 3.0, and for varying temperature T_e represented by dependence (21) (look Append. D).

To approximate the increase in the temperature of rubbing surfaces with time, we can use the exponential dependence, which makes it possible to take into account these peculiarities,

$$T_e(t) = T_e^0\left(1-\exp(-Ct)\right), \qquad (21)$$

where T_e^0 is the value of T_e being stabilized and C is the constant determining the rate of temperature increase at the initial stage.

Figure 4 shows the solution $\sigma(t)$ of the Langevin equation taking formula (21) into account. The upper part shows dependence (21). In accordance with the phase diagram (see Fig. 1), point *A* corresponds to the temperature of transition from dry friction (DF) to stick-slip (SS) regime. At the temperature depicted by point *B*, a further transition to sliding friction (SF) takes place. The lower part shows the evolution of stresses $\sigma(t)$, where three temporal regions are singled out: region I (DF), II (SS), and III (SF). In the first region, the friction force assumes the maximal value or increases; in the second region, stick-slip motion takes place; while in the third region, the friction force decreases. The figure is in good agreement with experimental data obtained in [11, 12].

3 VARIANTS OF ASSIGNMENT

Realize modeling of random process and stationary probability density for given models.

1. Describe procedure of numerical solution of the Langevin equation.
2. Create program for construction of time series for stochastic variable.
3. Create program for determination of probability distribution of stochastic variable realization based on constructed time series.

Variant 1

For Malthus-Verhyulst population dynamics model at values of noise intensity at which the boundary nature is changed of diffusion process and noise induced phase is formed. The Langevin equation is

$$\dot{x} = \alpha x - x^2 + \sigma x \xi(t) = f(x) + g(x)\xi(t),$$
$$\langle \xi(t) \rangle = 0, \langle \xi(t)\xi(t') \rangle = \delta(t - t').$$

4. Construct numerically probability distributions. Show that they coincide with analytical solution.

Variant 2

For genetic model at different values of noise intensity for illustration of self-organization effect. The Langevin equation is

$$\dot{x} = \frac{1}{2} - x + \alpha x(1-x) + \frac{\sigma^2}{2} x(1-x)(1-2x) + \sigma x(1-x)\xi(t),$$
$$\langle \xi(t) \rangle = 0, \ \langle \xi(t)\xi(t') \rangle = \delta(t - t').$$

4. Construct numerically probability distributions. Show that they coincide with analytical solution.
5. Determine the influence of parameter α and provide conclusion.

Variant 3

For arbitrary amplitude model the Langevin equation is

$$\dot{x} = f(x) + g(x)\xi(t), \quad f(x) = -\partial V(x)/\partial x, \ V(x) = \frac{A}{2}x^2 + \frac{B}{4}x^4,$$
$$A = \alpha(T - T_c), \ g(x) = 2^{1/2} x^a, \quad 0 \le a \le 1, \ \langle \xi(t) \rangle = 0,$$
$$\langle \xi(t)\xi(t') \rangle = \delta(t - t').$$

4. Numerically construct probability distributions for $T < T_c$ and $T > T_c$. Show that they coincide with analytical solution.
5. Show that at the index of multiplicative function $a > 1/2$ the absorbing state appears.

Variant 4

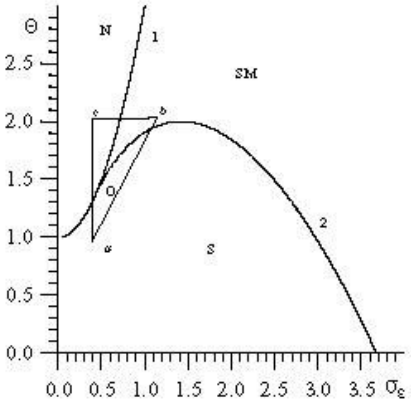
For Lorenz-Haken system in different domains of phase diagram considering control parameter noise. The Langevin equation is

$$\dot{\eta} = f(\eta) + \sigma_\varepsilon g_\varepsilon(\eta)\xi(t), \quad f(\eta) = -\partial V(\eta)/\partial\eta,$$

$$V(\eta) = \frac{1}{2}\left[\eta^2 - \Theta\ln\left(1+\eta^2\right)\right], \ \Theta = \frac{\varepsilon_0}{\varepsilon_c},$$

$$g_\varepsilon(\eta) = \eta\left(1+\eta^2\right)^{-1}, \ \left\langle\xi(t)\right\rangle = 0, \ \left\langle\xi(t)\xi(t')\right\rangle = \delta(t-t').$$

Phase diagram:



4. Construct numerically probability distributions. Show that they coincide with analytical solution.

REFERENCES

1. A. V. Khomenko, Phys. Lett. A **329**, 140 (2004).
2. A. V. Khomenko and I. A. Lyashchenko, Zh. Tekh. Fiz. **75** (11), 17 (2005) [Tech. Phys. **50**, 1408 (2005)].
3. A. V. Khomenko and O. V. Yushchenko, Phys. Rev. E **68**, 036110 (2003).
4. G. Luengo, J. Israelachvili, and S. Granick, Wear **200**, 328 (1996).
5. H. Romero Aldo, J. M. Sancho, and K. Lindenberg, Fluctuation and Noise Lett. **2**, L79 (2002).
6. H. Haken, Information and Self-Organization: A Macroscopic Approach to Complex Systems (Springer, Berlin, 2000; URSS, Moscow, 2005).

7. C. W. Gardiner, Handbook of Stochastic Methods (Springer, Berlin, 1994; Nauka, Moscow, 1985).
8. H. William, Numerical Recipes in C: The Art of Scientific *Computing* (Cambridge Univ. Press, New York, 1992).
9. M. L. Gee, P. M. McGuiggan, J. N. Israelachvili, and A. M. Homola, J. Chem. Phys. **93**, 1895 (1990).
10. I. S. Aranson, L. S. Tsimring, and V. M. Vinokur, Phys. Rev. B **65**, 125402 (2002).
11. H. Yoshizawa and J. Israelachvili, J. Phys. Chem. **97**, 11300 (1993).
12. O. M. Braun and A. G. Naumovets, Surf. Sci. Rep. **60**, 79 (2006).

Appendix A

Program for construction of phase diagram in Fig.1

main.cpp

```
#include <stdio.h>
#include <conio.h>
#include <math.h>

#define ItMin 0.0//
#define ItMax 18.0//
#define dI_t 0.01//

#define sigmaMin 0.0//1.0
#define sigmaMax 10.0//2.0

#define g 0.8
#define I_eps 0.8
#define D 0.8

//-----------------------------------------------------
#define dsigma 0.001//
double ReS(double sigma,double I_t)
{
 double x,Te;
 x = 1.0+sigma*sigma;
 Te = (1.0-g)*x*x*x - 2.0*g*g*D*I_t*x +
4.0*g*g*D*(I_t-I_eps);
 Te = 2.0 + Te/(g*x*x);
 return(Te);

}
//-----------------------------------------------------

void main()
{
 FILE*NuL = fopen("NULL.dat","wt");
 FILE*MINf_1 = fopen("Min1.dat","wt");
// FILE*MAXf_1 = fopen("Max1.dat","wt");
// FILE*MINf_2 = fopen("Min2.dat","wt");
// FILE*MAXf_2 = fopen("Max2.dat","wt");

 double Te,sigma,I_t,Te_null;
```

```
 double one_POINT,two_POINT,three_POINT;
 short unsigned int begin=0;
// short unsigned int file_changeMIN=1,
file_changeMAX=1;
  I_t = ItMin;
 while(I_t < ItMax)
 {
    sigma = sigmaMin;
    begin=0;
//    file_changeMIN=1;
//    file_changeMAX=1;
     while(sigma < sigmaMax)
       {
     one_POINT = two_POINT;
     two_POINT = three_POINT;
     three_POINT = ReS(sigma,I_t);
  // printf("%lf\n",three_POINT);
      if(begin == 0)
         {
          one_POINT = ReS(sigma,I_t);
            sigma+=dsigma;
          two_POINT = ReS(sigma,I_t);
            sigma+=dsigma;
          three_POINT = ReS(sigma,I_t);
         //   sigma-=dsigma;
         //   sigma-=dsigma;
           begin = 1;
         }
        sigma+=dsigma;
      if(two_POINT < one_POINT && two_POINT <
three_POINT)
//      {
//        if(file_changeMIN==1)
          fprintf(MINf_1,"%lf
%lf\n",I_t,two_POINT);
//         break;
//        if(file_changeMIN==2)
//         fprintf(MINf_2,"%lf
%lf\n",I_t,two_POINT);
//        file_changeMIN++;
//      }
```

```
//    if(two_POINT > one_POINT && two_POINT >
three_POINT)
//      {
//        if(file_changeMAX==1)
//         fprintf(MAXf_1,"%lf
%lf\n",I_t,two_POINT);
//        if(file_changeMAX==2)
//         fprintf(MAXf_2,"%lf
%lf\n",I_t,two_POINT);
//        file_changeMAX++;
//      }
       }
   Te_null = 1.0+1.0/g+2.0*g*D*(I_t-2.0*I_eps);
  fprintf(NuL,"%lf  %lf\n",I_t,Te_null);
  I_t+=dI_t;
 }
/////////////////
// fclose(MAXf_1);
 fclose(MINf_1);
// fclose(MAXf_2);
// fclose(MINf_2);
 fclose(NuL);
}
```

Program for calculation of tricritical point

tricriti.cpp

```
#include<stdio.h>
#include<conio.h>
#define I_eps 0.8
#define g 0.8
#define D 0.8
void main()
{
clrscr();
double Te,It;
 Te = (2.0/3.0)*(1.0+2.0/g - 2.0*D*g*I_eps);
 It = (1/(6.0*g*D))*(1/g-1+8*D*g*I_eps);

 printf("\n\n Te=%lf \n It=%lf",Te,It);
 getch();
}
```

Appendix B

Program for calculation of time series in Fig. 2

main.cpp

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <conio.h>
#define pi (4.0*atan(1.0))
//////////////////////
#define t_ALL     5000.0
#define coll_razb 200000.0
#define delta_t (t_ALL/coll_razb)
//////////////////////
#define g 0.8
#define I_eps 0.8
#define D 0.8
//
#define I_sigma 0.1
#define I_T 1.0
#define Te 4.0

#define tau_sigma 1.0
///*****
#define sig_0 0.0
//////////////////////
double BoxMuller(void)
{
 double gg;
 static double r[2];
  for(int j=0; j<2; j++)
  {
   r[j] = random(RAND_MAX)/(RAND_MAX + 1.0);
  if(r[j] <= 1e-10)
          r[j] = 1e-10;}
  gg = sqrt(fabs(2.0*logl(r[0])))*cos(2.0*pi*r[1]);
 return(gg);
}
////////////////////////////////////
void main()
{
```

```
_setcursortype(_NOCURSOR);
FILE*file = fopen("res.dat", "wt");
randomize();
clrscr();

double ff, I, t=0.0, sigma1, sigma2, percent,
percent_new;

sigma1 = sig_0;

 for(;;)
  {
   if(t>=t_ALL) break;
     t += delta_t;
     ff = -sigma1*(1.0-g) - (g*(1.0-
Te/2.0)*2.0*sigma1)/(1.0+sigma1*sigma1);
     I = I_sigma +
((I_eps+I_T*sigma1*sigma1)*g*g)/(pow(1.0+sigma1*sigma
1,2.0));
     sigma2 = sigma1 + ff*delta_t/tau_sigma +
sqrt(2.0*D)*sqrt(I)*sqrt(delta_t)*BoxMuller()/tau_sig
ma;
//     if(sigma2<0) sigma2=0;
     ///Potential kak v stat'e ///
     //sigma2 = sigma1 +  delta_t*(sigma1-
sigma1*sigma1*sigma1 + BoxMuller()*sqrt(2.0*D));
//      if(t>1000.0)
   fprintf(file,"%lf %lf \n",t,sigma1);
      /////
      percent = 100.0*t/t_ALL;
       if(ceil(percent)!=ceil(percent_new))
     {
       gotoxy(1,1);
       printf("%2.0lf", ceil(percent));
       percent_new = percent;
     }
      /////
   sigma1 = sigma2;
  }
  fclose(file);

}
```

Auxiliary program
optima.cpp

```
#include<stdio.h>
#include<math.h>

void main()
{
double X,Y;
int i=0;
 FILE*f = fopen("sf.dat","rt");
  FILE*f1 = fopen("sf_new.dat","wt");
  while(!feof(f))
   {
    i++;
    fscanf(f,"%lf %lf",&X,&Y);
     if(i==30) {
      i=0;
      fprintf(f1,"%lf %lf\n",X,Y);
            }
   }
  fclose(f);
  fclose(f1);
}
```

Appendix C

Programs for probabilities calculation in Fig. 3

Program for probability calculation using Simpson method simpson.cpp

```
#ifndef simpson_H__
#define simpson_H__
#include "funk.h"
#include <stdio.h>
#include <conio.h>
//#include <math.h>
//-----------------------------------------------------
// For Integral Equation:
double INTEGR(double X)
 {
  double INTEGR;
  INTEGR = f(X)/I(X);
  return INTEGR;
 };
//---------------------
double simpson(double A, double B, double eps)
{
 double I1,I2;
 double h,m;
 double X; //po X vedetsja integrirovanije
 double N;
   N=4.0;
   I1 = 0.0;
 do
  {
//   if(kbhit()) break;
   h=(B-A)/N;
   I2=(INTEGR(A)+INTEGR(B))/2.0;
   X = A+h;
      do
       {
//   if(kbhit()) break;
     I2=I2+2.0*INTEGR(X)+INTEGR(X+h);
     X = X+2.0*h;
       }
```

```
      while(X<=B);
   I2 = 2.0*h*I2/3.0;
   N = 2.0*N;
   m = fabs(I1-I2);
   I1=I2;
  //printf("%lf\n",I1);
  }
 while(m>eps);
//  printf("int=%lf\n",I2);

 return I1;
};
//----------------
#endif
```

Auxiliary program for Simpson method
funk.cpp

```
#ifndef FUNK_H__
#define FUNK_H__
#include <math.h>
//-----------------------------------------------------
const double g = 0.8;
const double I_sigma = 0.1;
const double D = 0.8;
const double I_eps = 0.8;
const double I_t = 3.0;
const double Te = 0.5;

//-----------------------------------------------------
double I(double sigma)
{
 double I;
 I = I_sigma + (I_eps +
I_t*sigma*sigma)*g*g*pow(1.0+sigma*sigma,-2.0);
 return I;
};
//-------------------
double f(double sigma)
{
 double f;
```

```
 f= -(1.0-g)*sigma-2.0*g*(1.0-
Te/2.0)*sigma/(1.0+sigma*sigma);
 return f;
};
//--------------------
#endif
```

Program for normalization
norm.cpp

```
#include <stdio.h>

#define s_begin -21.0
#define s_end 21.0

void main()
{
double Z,Int_SUM,x,y,x1,y1;
int i=0;
Z=0;
  FILE*f_Z=fopen("ver.dat","rt");
  FILE*Z__=fopen("Z_norm.dat","wt");
    while(!feof(f_Z))
    {
     fscanf(f_Z,"%lf %lf\n",&x1,&y1);
      if(i==0) {
      i=1;
      fscanf(f_Z,"%lf %lf\n",&x,&y);
            }
     Z += ((y+y1)/2.0)*(x1-x);
     x = x1;
     y = y1;
    }
    fprintf(Z__,"Z=%2.20lf",Z);
    fclose(Z__);

//Otkrojem fail dlia normirovki:
  FILE*out = fopen("ver.dat","rt");
  FILE*in = fopen("VER_nor.dat","wt");
   while(!feof(out))
    {
     fscanf(out,"%lf %lf\n",&x,&y);
```

```
       if(x>=s_begin && x<=s_end)
     fprintf(in,"%2.8lf %2.20lf\n",x,y/Z);
    }
  fclose(in);
  fclose(out);
  printf("done");
//
}
```

Program for probability density calculation using time series realized.cpp

```
/*
#define t_ALL     1000000.0
#define coll_razb 100000000.0
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <conio.h>
#define pi (4.0*atan(1.0))
//////////////////////
#define t_ALL     1000000.0
#define coll_razb 100000000.0
#define delta_t (t_ALL/coll_razb)
//////////////////////
#define g 0.8
#define I_eps 0.8
#define D 0.8
#define I_sigma 0.1
#define I_T 6.0
#define Te 2.5
///*****
#define sig_0 0.0
//////////////////////
double BoxMuller(void)
{
 double gg;
 static double r[2];
  for(int j=0; j<2; j++)
  {
   r[j] = random(RAND_MAX)/(RAND_MAX + 1.0);
  if(r[j] <= 1e-10)
```

```
          r[j] = 1e-10;}
  gg = sqrt(fabs(2.0*logl(r[0])))*cos(2.0*pi*r[1]);
 return(gg);
}
////////////////////////////////////
void main()
{
_setcursortype(_NOCURSOR);
FILE*file = fopen("res.dat", "wt");
FILE*MAXMIN = fopen("MaxMin.dat", "wt");
randomize();
clrscr();

double ff, I, t=0.0, sigma1, sigma2, percent,
percent_new;
double MAX_tr = 0.0, MIN_tr = 0.0;

sigma1 = sig_0;

 for(;;)
  {
   if(t>=t_ALL) break;
     t += delta_t;
     ff = -sigma1*(1.0-g) - (g*(1.0-
Te/2.0)*2.0*sigma1)/(1.0+sigma1*sigma1);
     I = I_sigma +
((I_eps+I_T*sigma1*sigma1)*g*g)/(pow(1.0+sigma1*sigma
1,2.0));
     sigma2 = sigma1 + ff*delta_t +
sqrt(2.0*D)*sqrt(I)*sqrt(delta_t)*BoxMuller();
     ///Potential kak v stat'e ///
     //sigma2 = sigma1 +  delta_t*(sigma1-
sigma1*sigma1*sigma1 + BoxMuller()*sqrt(2.0*D));
//      if(t>1000.0)
      if(sigma1<MIN_tr) MIN_tr = sigma1;
      if(sigma1>MAX_tr) MAX_tr = sigma1;

   fprintf(file,"%lf %lf \n",t,sigma1);
      /////
      percent = 100.0*t/t_ALL;
       if(ceil(percent)!=ceil(percent_new))
     {
       gotoxy(1,1);
```

```
       printf("%2.0lf", ceil(percent));
       percent_new = percent;
     }
      /////
   sigma1 = sigma2;
  }
  fprintf(MAXMIN,"%lf %lf", MAX_tr, MIN_tr);
  fclose(file);
  fclose(MAXMIN);
}
```

Program for probability plotting
ver_plot.cpp

```
#include<stdio.h>
#include<dir.h>
#include<conio.h>
#include<math.h>
#include<stdlib.h>
///////////
////////////////////////////////
#define pi 4*atan(1) //Pi

#define DX 1000 //Kolichestvo otrezkov, na kotorije
delitsja ordinata
///////////////
double OKRUGL(double XX);
///////////////
void main()
{
 clrscr();
 randomize();
 double MAX_tr, MIN_tr, dx, t=0.0, x,
Num_of_Interval;
 double i, COLL;

 double ff, I;
 FILE *f, *f_ver;
 MAX_tr = 0.0; MIN_tr = 0.0; COLL = 0.0;
////Schitajem verojatnost'////

/* f = fopen("res.dat", "rt");
   while(!feof(f))
```

```
    {
     fscanf(f,"%lf %lf\n",&t,&x);
      if(x<MIN_tr) MIN_tr = x;
      if(x>MAX_tr) MAX_tr = x;
    }
     fclose(f);*/
//
   f = fopen("MaxMin.dat", "rt");
     fscanf(f,"%lf %lf", &MAX_tr, &MIN_tr);
   fclose(f);
////////////////////////////////

  dx = (MAX_tr-MIN_tr)/DX; //Dlinna otrezka, na
kotorije delitsja ordinata
  printf("dx = %lf\n",dx);
   double far *MASS = new double[DX+1];
for(i=0;i<DX+1;i++) MASS[i] = 0.0;
      f = fopen("res.dat", "rt");

     while(!feof(f))
      {
//      gotoxy(10,10);
//       printf("%lf",Num_of_Interval);
       fscanf(f,"%lf %lf", &t, &x);
       Num_of_Interval = (x-MIN_tr)/dx;
        COLL++;
       MASS[OKRUGL(Num_of_Interval)]+=1.0;
//     printf("%d\n",int(OKRUGL(Num_of_Interval)));
       //getch();
      }
      fclose(f);

   f_ver = fopen("ver.dat","wt");
     for(i=0;i<DX;i++)
     if(MASS[i]!=0) //Ubiraem lishnije nulevije
tochki
      fprintf(f_ver,"%lf %lf\n", MIN_tr+i*dx-
dx/2.0,MASS[i]/COLL);
   fclose(f_ver);

 delete[]MASS;
 printf("\n\nDONE!!!");
////////////////////////////////////
```

```
}
////////////////////////////////////
double OKRUGL(double XX) //Funkchija dlja okruglenija
chisla
{
  double fraction, integer, RES;
   fraction = modf(XX, &integer);
   RES = integer;
    if(fraction >= 0.5) RES = RES+1.0;
  return RES;
}
```

Appendix D

Programs for construction of graph in Fig. 4

Program for time series calculation
realized.cpp

```
#include "all.h"
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#define pi (4.0*atan(1.0))
//////////////////////
#define t_ALL     10000.0
#define coll_razb 200000.0
#define delta_t (t_ALL/coll_razb)
//////////////////////

#define tau_sigma 1.0
///*****
#define sig_0 0.0
//////////////////////
double BoxMuller(void)
{
 double gg;
 static double r[2];
  for(int j=0; j<2; j++)
  {
   r[j] = random(RAND_MAX)/(RAND_MAX + 1.0);
  if(r[j] <= 1e-10)
          r[j] = 1e-10;}
  gg = sqrt(fabs(2.0*logl(r[0])))*cos(2.0*pi*r[1]);
 return(gg);
}
////////////////////////////////////
void main()
{
_setcursortype(_NOCURSOR);
FILE*file = fopen("res.dat", "wt");
randomize();
clrscr();
```

```
double ff, I, t=0.0, sigma1, sigma2, percent,
percent_new;
sigma1 = sig_0;
 for(;;)
  {
   if(t>=t_ALL) break;
     t += delta_t;
     ff = -sigma1*(1.0-g) - (g*(1.0-
Te(t)/2.0)*2.0*sigma1)/(1.0+sigma1*sigma1);
     I = I_sigma +
((I_eps+I_T*sigma1*sigma1)*g*g)/(pow(1.0+sigma1*sigma
1,2.0));
     sigma2 = sigma1 + ff*delta_t/tau_sigma +
sqrt(2.0*D)*sqrt(I)*sqrt(delta_t)*BoxMuller()/tau_sig
ma;
   fprintf(file,"%lf %lf \n",t,sigma1);
      /////
      percent = 100.0*t/t_ALL;
       if(ceil(percent)!=ceil(percent_new))
     {
       gotoxy(1,1);
       printf("%2.0lf", ceil(percent));
       percent_new = percent;
     }
      /////
   sigma1 = sigma2;
  }
  fclose(file);
}
```

Auxiliary program
optima.cpp

```
#include<stdio.h>
#include<math.h>
void main()
{
double X,Y;
int i=0;
 FILE*f = fopen("RES.dat","rt");
  FILE*f1 = fopen("RES_new.dat","wt");
  while(!feof(f))
```

```
   {
    i++;
    fscanf(f,"%lf %lf",&X,&Y);
     if(i==30) {
      i=0;
      fprintf(f1,"%lf %lf\n",X,Y);
            }
   }
  fclose(f);
  fclose(f1);
}
```

Program for critical points calculation (T_e(t).cpp)

```
#include "all.h"
#include<stdio.h>
void main()
{
FILE*f = fopen("Te(t).dat", "wt");
  for(int t=0; t<10000; t++)
    {
     fprintf(f,"%d %lf\n", t, Te(t));
    }
fclose(f);}
```

Program for temperature calculation (points.cpp)

```
#include "all.h"
#include<stdio.h>
void main()
{
double Te;
FILE*f = fopen("points.dat", "wt");
Te = 1.0 + 1.0/g + 2.0*g*D*(I_T - 2*I_eps);
fprintf(f,"MAX: Te = %lf", Te);
fclose(f);
}
```

Навчальне видання

МЕТОДИЧНІ ВКАЗІВКИ

для практичних робіт
із дисципліни «Нелінійні процеси та моделі»
з теми «Моделювання випадкового процесу»
для студентів спеціальності 113 «Прикладна математика» денної форми навчання

Відповідальний за випуск І. В. Коплик
Редактор Л. В. Штихно
Комп'ютерне верстання О. В. Хоменка, А. М. Заскоки

Підписано до друку 12.01.2018, поз.
Формат 60х84/16. Ум. друк. арк. 2,09.Обл.-вид. арк. 2,46. Тираж 40 пр. Зам. №
Собівартість видання грн. к.

Видавець і виготовлювач
Сумський державний університет,
вул. Римського-Корсакова, 2, м.Суми, 40007
Свідоцтво суб'єкта видавничої справи ДК № 3062 від 17.12.2007.