



LUND UNIVERSITY

Simulation of Non-linear Stochastic Differential Equations

Razevig, Vsevolod D.

1977

Document Version:

Publisher's PDF, also known as Version of record

[Link to publication](#)

Citation for published version (APA):

Razevig, V. D. (1977). *Simulation of Non-linear Stochastic Differential Equations*. (Technical Reports TFRT-7120). Department of Automatic Control, Lund Institute of Technology, Lund University.

Total number of authors:

1

General rights

Unless other specific re-use rights are stated the following general rights apply:

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal

Read more about Creative commons licenses: <https://creativecommons.org/licenses/>

Take down policy

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

LUND UNIVERSITY

PO Box 117
221 00 Lund
+46 46-222 00 00

SIMULATION OF NON-LINEAR STOCHASTIC
DIFFERENTIAL EQUATION

VSEVOLOD D. RAZEVIĆ

Department of Automatic Control
Lund Institute of Technology
May 1977

Dokumentutgivare
Lund Institute of Technology
Handläggare Dept of Automatic Control
06T0
Författare
08T0
Vsevolod D. Razevig

Dokumentnamn
06T4
REPORT
Utgivningsdatum
06T4
May 1977

Dokumentbeteckning
06T6
LUTFD2/(TFRT-7120)/1-52/(1977)
Ärendebeteckning
06T6

10T4

Dokumenttitel och undertitel

13T0
Simulation of Non-linear Stochastic Differential Equations

Referat (sammandrag)

26T0
This paper describes a numerical technique to solve non-linear stochastic differential equations of Itô and Stratonovich type. We consider Euler, fourth-order Runge-Kutta (R-K) schemes, and other schemes with intermediate accuracy. For the purpose of investigating the convergence of numerical solutions and to apply variable integration step length techniques the special Wiener process generator was developed. The main result of the paper is the FORTRAN program combining Euler and R-K methods both with constant and variable integration step lengths. In an example the accuracy of these methods is compared.

This work was supported by a scholarship from the Swedish Institute.

Referat skrivet av

Author

Förslag till ytterligare nyckelord

44T0
stochastic differential equation, numerical solution

Klassifikationssystem och -klass(er)

50T0

Indextermer (ange källa)

52T0

Omfång

58 pages

Övriga bibliografiska uppgifter

56T2

Språk

English

Sekretessuppgifter

60T0

ISSN

60T4

ISBN

60T6

Dokumentet kan erhållas från

Department of Automatic Control
Lund Institute of Technology
Box 725, S-220 07 LUND 7, SWEDEN

Mottagarens uppgifter

62T4

Pris

66T0

DOKUMENTATABLAD enligt SIS 62 10 12

SIS-
DB 1

Blankett LU 11:25 1976-07

SIMULATION OF NON-LINEAR STOCHASTIC DIFFERENTIAL
EQUATIONS

Vsevolod D. Razevig

May 1977

Department of Automatic Control
Lund Institute of Technology

Table of Contents

	Page
1. Introduction	4
2. Difference Schemes to Obtain Ito and Stratonovich Solutions	6
2.1 Preliminary	6
2.2 Stratonovich Solution of One-dimensional Equations	7
2.3 Ito Solution of One-dimensional Equations	10
2.4 Generalization to Many-dimensional Equations	12
3. Wiener Process Generator	14
3.1 Algorithm	14
3.2 Comparison of two Gaussian Random Number Generators	16
4. Variable Integration Step Size Technique (Step Doubling and Halving)	19
5. Description of the FORTRAN Program	21
5.1 Main Program	21
5.2 Standard Subroutines	22
5.3 User Subroutines	22
6. Examples	32
7. Acknowledgements	48
8. References	49
9. Appendix	51

1. INTRODUCTION

This report discusses the problem of digital simulation of the vector nonlinear stochastic differential equation

$$\frac{dx}{dt} = a(x(t), t) + b(x(t), t) \xi(t), \quad (1)$$

$$t \in [t_0 = t_{\min}, t_{\max}]$$

where $x = (x_1, x_2, \dots, x_n)$ is a $n \times 1$ state variables vector, $a(x, t) = \| a_i \|$ is a $n \times 1$ vector functions of x and t , $b(x, t) = \| b_{ij} \|$ is a $n \times m$ matrix function. $a(x, t)$ and $b(x, t)$ are satisfied to the Lipschitz conditions. Random forcing $\xi(t) = \frac{d}{dt}w(t)$ is a $m \times 1$ vector of white noises, $w(t)$ is a $m \times 1$ standard Wiener process

$$w_i(t_0) = 0, \quad i = \overline{1, m},$$

$$E\{w_i(t)\} = 0, \quad (2)$$

$$E\{w_i(t_2)w_j(t_1)\} = \delta_{ij}|t_2 - t_1|.$$

The initial condition of state variables equals $x(t_0)$.

There are few papers on numerical integration of the stochastic differential equation (1), see [1-9, 14, 15]. The theoretical estimations of accuracy are discussed only in [2] and [9]. The articles [3, 5, 8] present the experiment experience. The papers [5, 6, 14, 15] are devoted to numerical schemes providing only the statistical convergence, i.e. the convergence of the sample moments. The sample path convergence is discussed in the Ph.D. thesis [2], the book [4, p.186], and the paper [9].

This report deals with:

- (a) the application of the Picard iterations [1, p. 279] to obtain several integration schemes (not only Euler and Runge-Kutta schemes) which provide a sample path convergence;

- (b) the development of a special Wiener process generator applicable to sample path convergence investigations and to numerical schemes with variable integration step lengths;
- (c) the development of routines to integrate the n-dimensional stochastic differential equation (1) in Itô and Stratonovich senses both with constant and variable integration step lengths. The routines are written for FORTRAN-4 compiler at computer PDP-15 (versus FOR at the Department of Automatic Control).

2. DIFFERENCE SCHEMES TO OBTAIN ITÔ AND STRATONOVICH SOLUTIONS

2.1 Preliminary

In the theoretical approach it is more convenient to work with the integral equation equivalent to the differential equation (1):

$$x(t) = x(t_0) + \int_{t_0}^t a(x(s), s) ds + \int_{t_0}^t b(x(s), s) dw(s). \quad (3)$$

To simplify the calculations we consider equation (3) to be the one-dimensional one. The extension to the n-dimensional equation is rather clear, some necessary remarks are pointed out in Section 2.4.

We use the Picard iterations [1, p. 279] to obtain difference schemes for solving equation (3):

$$x_t^{[r+1]} = x_{t_0} + \int_{t_0}^t a(x_s^{[r]}, s) ds + \int_{t_0}^t b(x_s^{[r]}, s) dw_s \quad (4)$$

where $x_t = x(t)$, $w_t = w(t)$, index in square brackets indicates the number of iteration.

Let us choose the zero approximation as $x_t^{[0]} = x_{t_0}$. Then, according to (4), the first approximation is

$$\begin{aligned} x_t^{[1]} &= x_{t_0} + \int_{t_0}^t a(x_{t_0}, s) ds + \int_{t_0}^t b(x_{t_0}, s) dw_s \approx \\ &\approx x_{t_0} + a(x_{t_0}, t_0) \Delta t + b(x_{t_0}, t_0) \Delta w_t \end{aligned} \quad (5)$$

where $\Delta t = t - t_0$, $\Delta w_t = w(t) - w(t_0)$.

Putting (5) into (4) we get the second approximation:

$$\begin{aligned}
 x_t^{[2]} &= x_{t_0} + \int_{t_0}^t a \left(x_{t_0} + a(x_{t_0}, t_0) \cdot (s-t_0) + b(x_{t_0}, t_0) \cdot (w(s) - w(t_0)), s \right) ds \\
 &\quad + \int_{t_0}^t b \left(x_{t_0} + a(x_{t_0}, t_0) \cdot (s-t_0) + b(x_{t_0}, t_0) \cdot (w(s) - w(t_0)), s \right) dw_s \approx \quad (6) \\
 &\approx x_{t_0} + a \cdot \Delta t + (a'_t + a \cdot a'_x) \frac{(\Delta t)^2}{2} + a'_x b \int_{t_0}^t \Delta w_s ds + \dots + \\
 &\quad + b \cdot \Delta w_t + (b'_t + a \cdot b'_x) \int_{t_0}^t \Delta s dw_s + b \cdot b'_x \int_{t_0}^t \Delta w_s dw_s + \dots
 \end{aligned}$$

In this expression we used the notations

$$\begin{aligned}
 a &= a(x_{t_0}, t_0), \quad b = b(x_{t_0}, t_0), \\
 a'_t &= \left. \frac{\partial}{\partial t} a(x, t) \right|_{x_{t_0}, t_0} \quad \text{etc.}
 \end{aligned}$$

Expression (6) contains stochastic integrals. Therefore the results of calculations depend on the definition of the stochastic integral. Consider below both the Itô and the Stratonovich solutions of the stochastic differential equation (1).

2.2 Stratonovich Solution

It can be shown [1] that the first approximation (5) converges to the Itô solution of (1). The simplest difference scheme to find the Stratonovich solution runs from the

second approximation (6) retaining only the terms order Δt and $(\Delta w_t)^2$:

$$x_t = x_{t_0} + a \cdot \Delta t + \frac{1}{2} b b'_x (\Delta w_t)^2 \quad (7)$$

where we use the well-known Stratonovich integral [10]

$$\int_{t_0}^t \Delta w_s \, dw_s = \frac{1}{2} (\Delta w_t)^2. \quad (8)$$

A more accurate difference scheme follows from the full expression for the second approximation (6). This expression contains two stochastic integrals

$$\int_{t_0}^t \Delta w_s \, ds \quad \text{and} \quad \int_{t_0}^t \Delta s \, dw_s$$

which are the same both in the Itô and Stratonovich senses. Therefore we can use the Itô lemma [6]

$$\Delta w_t \cdot \Delta t = \int_{t_0}^t \Delta s \, dw_s + \int_{t_0}^t \Delta w_s \, ds \quad (9)$$

and calculate only one of them, for example

$$\eta = \int_{t_0}^t \Delta w_s \, ds. \quad (10)$$

The integral η (10) can't be expressed in the closed form. So, we find the optimal estimate in the mean square sense

$$\hat{\eta} = E \{ \eta \mid w(t), w(t_0) \} = \frac{\Delta t \cdot \Delta w_t}{2}. \quad (11)$$

The estimate $\hat{\eta}$ converges to η as Δt is refined for every realization of $w(t)$.

Hence we have got a difference scheme more accurate than (7):

$$\begin{aligned}
x_t = x_{t_0} + a \cdot \Delta t + b \cdot \Delta w_t + \frac{1}{2} b \cdot b'_x \cdot (\Delta w_t)^2 + \\
+ (a'_t + a \cdot a'_x) \frac{(\Delta t)^2}{2} + (a'_x \cdot b + b'_t + a \cdot b'_x) \frac{\Delta t \cdot \Delta w_t}{2}
\end{aligned} \quad (12)$$

providing Stratonovich solution of equation (1).

We can't estimate other stochastic integrals of higher order such as $\int_{t_0}^t (\Delta w_s)^2 ds$ and so on. But we can partly improve the accuracy retaining the higher derivatives of function $a(x,t)$ at the second approximation (6). Thus we get the scheme analogous to the fourth-order Runge-Kutta (R-K) scheme for deterministic equations [2,11]

$$x_t = x_{t_0} + \frac{1}{6}(K_1 + 2K_2 + 2K_3 + K_4) \quad (13)$$

where

$$K_1 = a(x_{t_0}, t_0) \Delta t + b(x_{t_0}, t_0) \Delta w_t,$$

$$K_2 = a\left(x_{t_0} + \frac{K_1}{2}, t_0 + \frac{\Delta t}{2}\right) \Delta t + b\left(x_{t_0} + \frac{K_1}{2}, t_0 + \frac{\Delta t}{2}\right) \Delta w_t,$$

$$K_3 = a\left(x_{t_0} + \frac{K_2}{2}, t_0 + \frac{\Delta t}{2}\right) \Delta t + b\left(x_{t_0} + \frac{K_2}{2}, t_0 + \frac{\Delta t}{2}\right) \Delta w_t,$$

$$K_4 = a(x_{t_0} + K_3, t_0 + \Delta t) \Delta t + b(x_{t_0} + K_3, t_0 + \Delta t) \Delta w_t,$$

the Wiener process increment $\Delta w_t = w(t) - w(t_0)$.

When the coefficient $b(x,t)$ in equation (1) is rather small, scheme (13) has advantages compared to scheme (12). But for large $b(x,t)$ both schemes have approximately the same accuracy. A similar phenomenon was at first mentioned in [2] and then in [4].

The Runge-Kutta method (13) is the most convenient method to provide the Stratonovich solution of equation (2). Compared with schemes (7) and (12) this method has the better accuracy and demands to calculate only the coefficients $a(x,t)$ and $b(x,t)$, not their higher derivatives.

Therefore just method (13) was used in the FORTRAN program (see Section 5) to obtain the Stratonovich solution.

2.3 Itô Solution of One-dimensional Equation

As it was mentioned above, the first approximation (5)

$$x_t = x_{t_0} + a \cdot \Delta t + b \cdot \Delta w_t \quad (5')$$

converges to the Itô solution of equation (1). We call this (5') as the Euler scheme has expanded to the stochastic differential equations.

Putting the Itô integral [10]

$$\int_{t_0}^t \Delta w_s dw_s = \frac{1}{2} [(\Delta w_t)^2 - \Delta t]$$

into the expression for the second approximation (6) and retaining only the terms order Δt and $(\Delta w_t)^2$ we get a more accurate scheme than (5'),

$$x_t = x_{t_0} + a \cdot \Delta t + \frac{1}{2} b \cdot b'_x [(\Delta w_t)^2 - \Delta t] \quad (14)$$

Note that if we put the approximate equality $(\Delta w_t)^2 \approx \Delta t$ into the right-hand side of (14) we'll get the previous scheme (5').

The difference scheme similar to (12) is

$$\begin{aligned} x_t = x_{t_0} + a \cdot \Delta t + b \cdot \Delta w_t + \frac{1}{2} b \cdot b'_x [(\Delta w_t)^2 - \Delta t] + \\ + (a'_t + a \cdot a'_x) \frac{(\Delta t)^2}{2} + (a'_x \cdot b + b'_t + a \cdot b'_x) \frac{\Delta t \cdot \Delta w_t}{2} . \end{aligned} \quad (15)$$

Scheme (15), providing the Itô solution, differs from the Stratonovich solution scheme (12) only by the term $\frac{1}{2} b \cdot b'_x \cdot \Delta t$. Hence the R-K scheme to obtain the Itô solution of equation (1) differs from expression (13) by the same term:

$$x_t = x_{t_0} + \frac{1}{6} (K_1 + 2K_2 + 2K_3 + K_4) - \frac{1}{2} b \cdot b'_x \cdot \Delta t \quad (16)$$

where

$$\begin{aligned}
 K_1 &= a(x_{t_0}, t_0) \Delta t + b(x_{t_0}, t_0) \Delta w_t, \\
 K_2 &= a(x_{t_0} + \frac{K_1}{2}, t_0 + \frac{\Delta t}{2}) \Delta t + b(x_{t_0} + \frac{K_1}{2}, t_0 + \frac{\Delta t}{2}) \Delta w_t, \\
 K_3 &= a(x_{t_0} + \frac{K_2}{2}, t_0 + \frac{\Delta t}{2}) \Delta t + b(x_{t_0} + \frac{K_2}{2}, t_0 + \frac{\Delta t}{2}) \Delta w_t, \\
 K_4 &= a(x_{t_0} + K_3, t_0 + \Delta t) \Delta t + b(x_{t_0} + K_3, t_0 + \Delta t) \Delta w_t.
 \end{aligned}$$

Another way to get an R-K scheme to obtain the $It\hat{o}$ solution of equation (1) is to apply the standard R-K scheme (13) to the next Stratonovich equation equivalent to the $It\hat{o}$ one (1):

$$dx = [a(x, t) - \frac{1}{2} b(x, t) b'_x(x, t)] dt + b(x, t) dw(t). \quad (17)$$

Hence we get an R-K scheme like (13), applied to the equivalent Stratonovich equation (17):

$$x_t = x_{t_0} + \frac{1}{6} (K_1 + 2K_2 + 2K_3 + K_4) \quad (18)$$

where

$$\begin{aligned}
 K_1 &= [a(x_{t_0}, t_0) - \frac{1}{2} b b'_x(x_{t_0}, t_0)] \Delta t + b(x_{t_0}, t_0) \Delta w_t, \\
 K_2 &= [a(x_{t_0} + \frac{K_1}{2}, t_0 + \frac{\Delta t}{2}) - \frac{1}{2} b b'_x(x_{t_0} + \frac{K_1}{2}, t_0 + \frac{\Delta t}{2})] \Delta t + \\
 &\quad + b(x_{t_0} + \frac{K_1}{2}, t_0 + \frac{\Delta t}{2}) \Delta w_t, \\
 K_3 &= [a(x_{t_0} + \frac{K_2}{2}, t_0 + \frac{\Delta t}{2}) - \frac{1}{2} b b'_x(x_{t_0} + \frac{K_2}{2}, t_0 + \frac{\Delta t}{2})] \Delta t + \\
 &\quad + b(x_{t_0} + \frac{K_2}{2}, t_0 + \frac{\Delta t}{2}) \Delta w_t, \\
 K_4 &= [a(x_{t_0} + K_3, t_0 + \Delta t) - \frac{1}{2} b b'_x(x_{t_0} + K_3, t_0 + \Delta t)] \Delta t + \\
 &\quad + b(x_{t_0} + K_3, t_0 + \Delta t) \Delta w_t.
 \end{aligned}$$

Scheme (16) has some advantages compared to the standard R-K scheme (18) as it demands to calculate the term $\frac{1}{2} b b'_x \Delta t$ only

once per each integration step. At the same time it has drawbacks from the point of programming.

To solve the Itô equation (1) we use the Euler method (5) in the FORTRAN program (see Section 5). In order to apply the standard R-K method (18) it is preliminarily necessary to calculate the term $\frac{1}{2}b(x,t)b'_x(x,t)$ of the equivalent Stratonovich equation (17) analytically (see also next section, 2.4).

2.4 Generalization to Many-dimensional Equations

The generalization of the R-K scheme (13) and the Euler scheme (5') for equation (1) to be n-dimensional is rather clear. It is necessary to replace scalar values by vectors and matrices [11]. That is only a formal thing to do. As for the R-K scheme (18) for the equivalent Stratonovich equation (17), it is necessary to calculate the term $\frac{1}{2}bb'_x$. Consider three cases:

2.4.1. In the general case the scalar form of the Itô vector equation (1) is

$$\begin{cases} \frac{dx_1}{dt} = a_1(\vec{x}, t) + \sum_{j=1}^m b_{1j}(\vec{x}, t) \xi_j(t) \\ \vdots \\ \frac{dx_n}{dt} = a_n(\vec{x}, t) + \sum_{j=1}^m b_{nj}(\vec{x}, t) \xi_j(t). \end{cases} \quad (19)$$

Using the Stratonovich rule [10] we find that the scalar term $\frac{1}{2}bb'_x$ in equation (17) corresponds to the term

$$\frac{1}{2} \sum_{j=1}^n \sum_{k=1}^m \frac{\partial b_{ik}(\vec{x}, t)}{\partial x_j} b_{jn}(\vec{x}, t), \quad i = \overline{1, n}. \quad (20)$$

2.4.2. In the second case there is only one white noise excitation, $\xi(t)$:

$$\begin{cases} \frac{dx_1}{dt} = a_1(\vec{x}, t) + g_1(\vec{x}, t) \xi(t), \\ \vdots \\ \frac{dx_n}{dt} = a_n(\vec{x}, t) + g_n(\vec{x}, t) \xi(t) \end{cases} \quad (21)$$

so the term $\frac{1}{2} b b'_x$ corresponds to

$$\frac{1}{2} \sum_{j=1}^n \frac{\partial g_i(\vec{x}, t)}{\partial x_j} g_j(\vec{x}, t), \quad i = \overline{1, n}. \quad (22)$$

2.4.3. In the third case every scalar equation contains only one white noise:

$$\begin{cases} \frac{dx_1}{dt} = a_1(\vec{x}, t) + g_1(\vec{x}, t) \xi_1(t), \\ \vdots \\ \frac{dx_n}{dt} = a_n(\vec{x}, t) + g_n(\vec{x}, t) \xi_n(t) \end{cases} \quad (23)$$

so the term $\frac{1}{2} b b'_x$ corresponds to

$$\frac{1}{2} \frac{\partial g_i(\vec{x}, t)}{\partial x_i} g_i(\vec{x}, t), \quad i = \overline{1, n} \quad (24)$$

Remark. A difference scheme to solve the n-dimensional Itô equation (19), similar to (15), is shown in Appendix.

3. WIENER PROCESS GENERATOR

3.1 Algorithm

While digital simulating the stochastic differential equations it is necessary to check the accuracy and to be sure that the approximate solution converges to the exact one (we consider the convergence in the sample path sense). So, we are to repeat simulations with different integration step lengths taking the same realization of the vector Wiener process $\vec{w}(t)$. It means that the samples of an approximate Wiener process at any point t_r equal the values of finer approximation at that time. The same problem takes place at the variable integration step method (see Section 4).

To do this, let us choose the smallest integration step length

$$h_{\min} = \frac{t_{\max} - t_{\min}}{2^{K_{\max}}}. \quad (25)$$

The current step length equals

$$h = \frac{t_{\max} - t_{\min}}{2^K} \quad (26)$$

where $K = K_{\min}, K_{\min} + 1, \dots, K_{\max}$. The value of K is appointed by the user in the constant integration step length method or is automatically chosen in the variable integration step length method.

So, if we have a sample of one-dimensional standard Wiener process $w(t_r)$ at any point t_r and are to generate the next sample $w(t_{r+1})$ at the point $t_{r+1} = t_r + h_r$, where step h_r is equal to (26), we use the algorithm (see Fig. 1):

$$w(t_{r+1}) = w(t_r) + \sqrt{h_{\min}} \sum_{\ell=1}^{h/h_{\min}} \zeta_{J_r+\ell} \quad (27)$$

where ζ_i is Gaussian random numbers with zero mean and unite

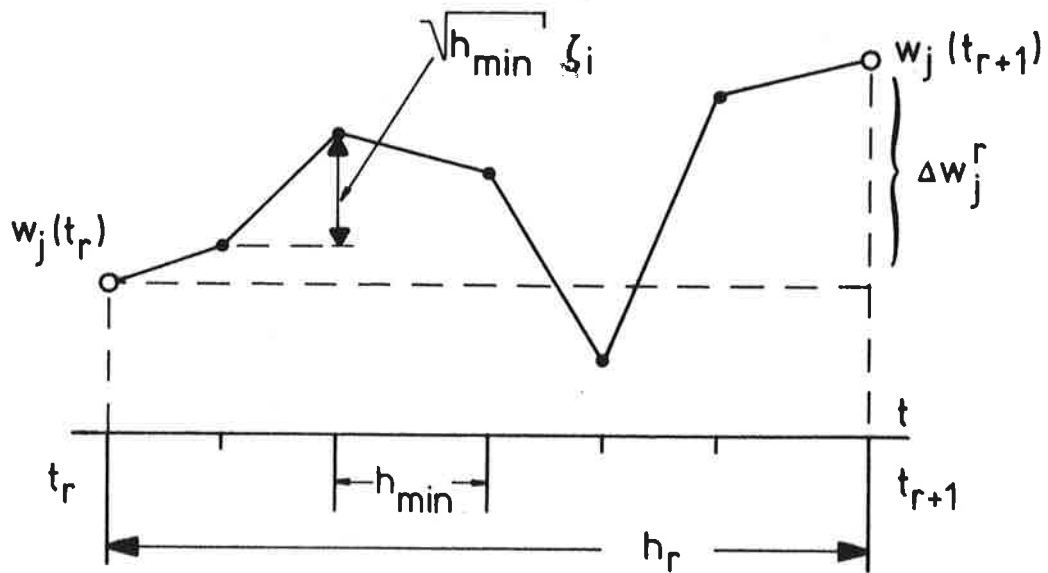


Fig. 1 - Wiener process generator.

variance, J_r is the number of the latest sample of the random number generator. The increment of one-dimensional Wiener process is equal to

$$\Delta w_r = w(t_{r+1}) - w(t_r) = \sqrt{h_{\min}} \sum_{\ell=1}^{h/h_{\min}} \zeta_{J_r+\ell} \quad (28)$$

Generating the m -dimensional Wiener process $\vec{w}(t) = (w_1(t), \dots, w_m(t))$ by one random number generator we use the algorithm

$$w_j(t_{r+1}) = w_j(t_r) + \sqrt{h_{\min}} \sum_{\ell=1}^{h/h_{\min}} \zeta_{J_r+(\ell-1)m+j} \quad (29)$$

where $j = 1, 2, \dots, m$.

Algorithm (29) was written in FORTRAN as

```

9 DO 11 J=1,M
11 R(J)=0.0
   DO 12 I=1,KR
   DO 12 J=1,M
   CALL MCNODI(NU,GS)
12 R(J)=R(J)+GS
   DO 13 J=1,M
   R(J)=R(J)*SHM
13 W(J)=W(J)+R(J)

```

(30)

where M is the number of scalar Wiener processes, $KR = h/h_{\min}$,
 $= 2^{K_{\max} - K}$, $SHM = \sqrt{h_{\min}}$, $MCNODI(NU,GS)$ is the Gaussian
 routine name, NU is an odd number, GS is $N(0,1)$ Gaussian
 random numbers, $R(J)$ is equal to the Wiener process
 increment $w_j(t_{r+1}) - w_j(t_r)$.

3.2 Comparison of two Gaussian Random Number Generators

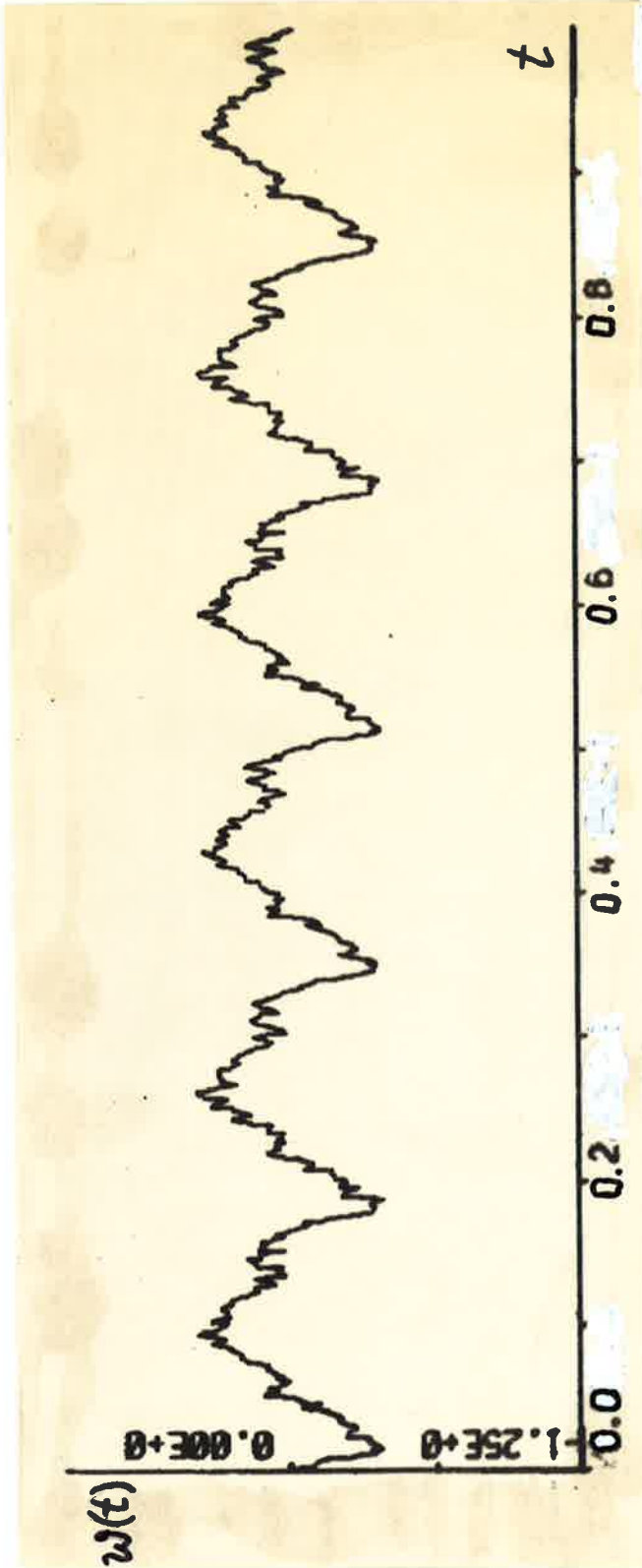
At first we tested algorithm (29) with a PDP-15 Gaussian
 random number routine $MCNODI$ like (30). It was obtained
 that this routine provides Wiener processes with periodical
 components, see Fig. 2a. We divided the time interval
 $t \in [0,1]$ on $2^{15} = 32\,768$ points and tested the $MCNODI$
 routine with several initial numbers NU (Fig. 2a corresponds
 to $NU = 9$) but at every realization a period of approximate-
 ly 5460 numbers was obtained.

To increase the period we used nonlinear transformation of
 two rectangular random variables [12, p.176]

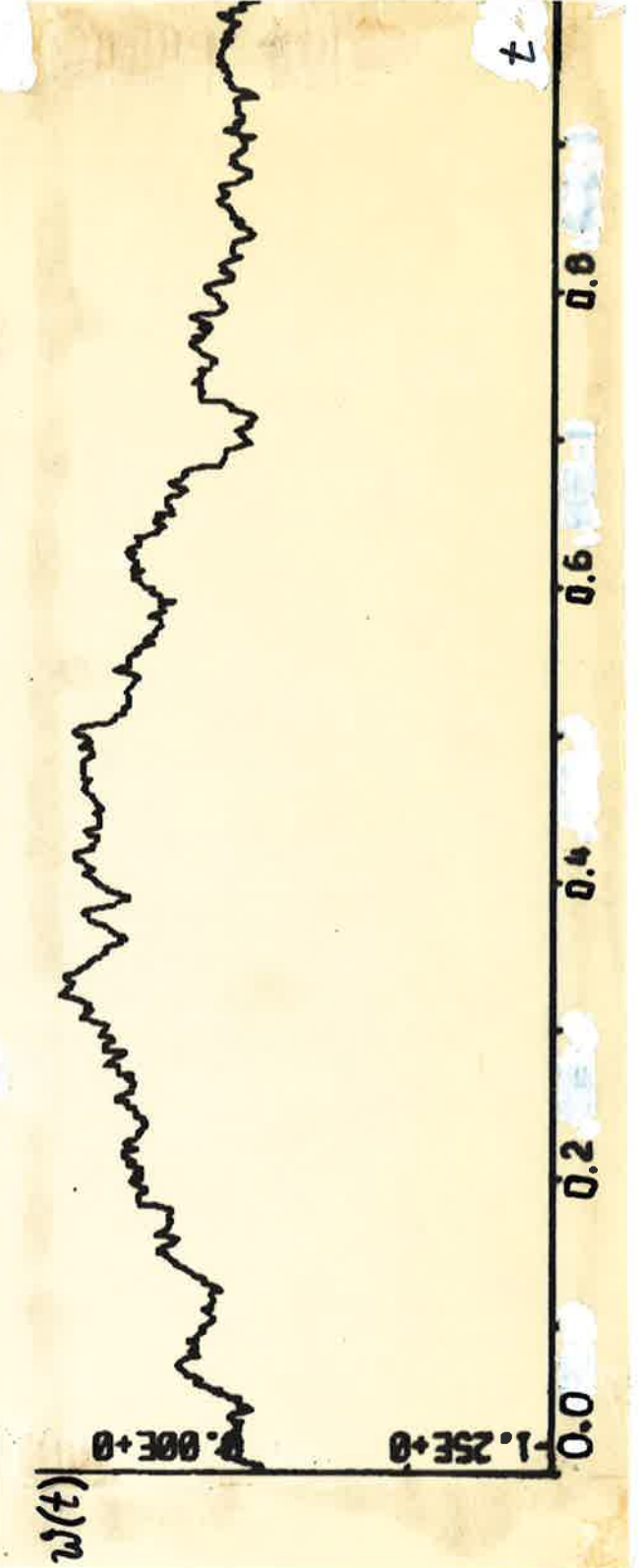
$$\begin{aligned} y_1 &= (-2 \ln \zeta_1)^{1/2} \cos 2\pi\zeta_2 \\ y_2 &= (-2 \ln \zeta_1)^{1/2} \sin 2\pi\zeta_2. \end{aligned} \quad (31)$$

Algorithm (31) transforms two independent random variables,
 ζ_1 and ζ_2 , rectangular over $[0,1]$ to two independent
 variables, y_1 and y_2 , from a Gaussian distribution with
 zero mean and unit variance.

To simplify the program we use only one random number, y_1 .
 Rectangular variables provides the routine $MCREDI$ which
 gives one zero sample per period. The FORTRAN program to
 generate the Wiener process (29) is



(a)



(b)

Fig.2. Wiener process: (a) MCNODI generator, (b) nonlinear transformation of rectangular random variables.

```

 9 DO 11 J=1,M
11 R(J)=0.0
   DO 12 I=1,KR
   DO 12 J=1,M
99 CONTINUE
   CALL MCREDI(NU,REC1)
   CALL MCREDI(NU,REC2)
   IF(REC1.EQ.0.0)GOTO 99
   GS=SQRT(-2.0*ALOG(REC1))*COS(PI2*REC2)
12 R(J)=R(J)+GS
   DO 13 J=1,M
   R(J)=R(J)*SHM
13 W(J)=W(J)+R(J)

```

(32)

The period of such a generator on PDP-15 is equal to $2^{17}-1 = 131071$. A realization of a Wiener process with initial number $NU=9$ is shown in Fig. 2b.

In conclusion, we point out the dependence between the mean value m and the standard deviation σ of the Gaussian generator (32) on N numbers of sampling:

N	m	σ
2	1.0080227554	1.3937898577
4	0.7538282275	1.0280693471
8	0.1870607361	1.2018311917
16	0.0475175977	1.2200787663
32	0.1173253767	1.0714786648
64	-0.0137966501	0.9706972837
128	-0.0495720468	1.0398858189
256	0.0026516137	1.0168921351
512	0.0438656453	0.9735938460
1024	0.0572415916	0.9886081666
2048	-0.0095007990	1.0021583736
4096	-0.0236054291	1.0073654949
8192	-0.0117221705	0.9965249449
16384	-0.0032938370	0.9994821548
32768	0.0002269202	1.0002311468

4. VARIABLE INTEGRATION STEP SIZE TECHNIQUE (STEP DOUBLING AND HALVING)

The estimation of local truncation error and integration step size control developed for ordinary differential equation [11] doesn't prove for stochastic ones. However, we have no alternative, as it is desirable to reduce the computation time applying the variable integration step length method. Therefore let's try to do this.

Each basic step of size h_r is done twice, once as two steps of size $h_r/2$ and once as one step of size h_r (see Fig. 3).

If the result of one step of size h_r is $x_{r+1}^{(1)}$, while the result of two steps of size $h_r/2$ is $x_{r+1}^{(2)}$, we have a local truncation error

$$\delta_{r+1} = \left| x_{r+1}^{(1)} - x_{r+1}^{(2)} \right|. \quad (33)$$

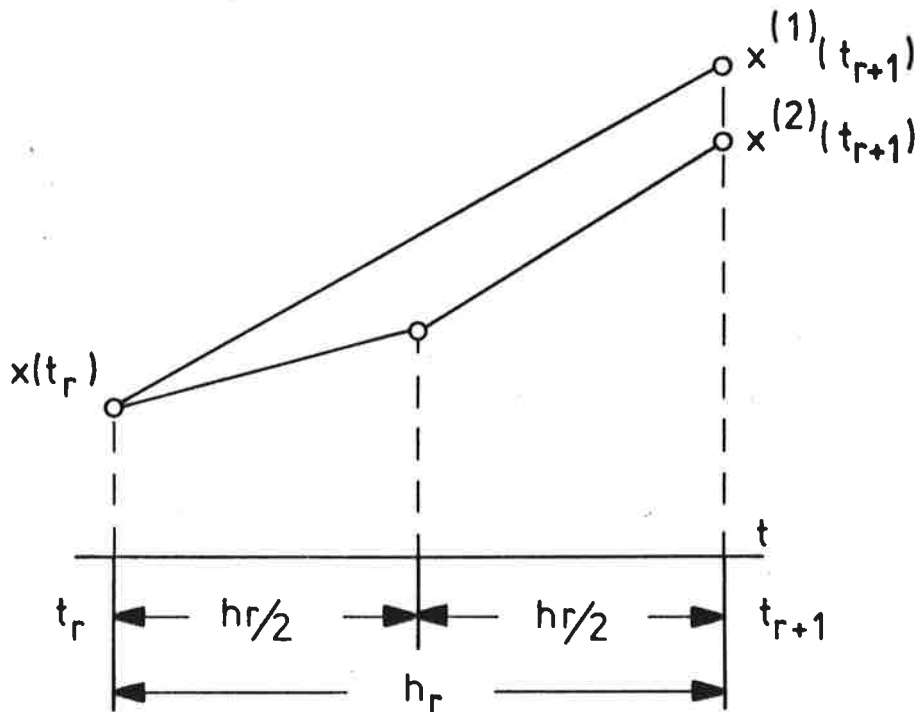


Fig. 3 - Variable step size.

In the n -dimensional case $\vec{x} = (x_1, x_2, \dots, x_n)$ is a vector, so

$$\delta_{r+1} = \max_i \left| x_i^{(1)}(t_{r+1}) - x_i^{(2)}(t_{r+1}) \right|. \quad (34)$$

Then compare δ_{r+1} with some limit error ε .

If $\delta_{r+1} > \varepsilon$, restart from point t_r halving the integration step length $h_r = h_r/2$. To get the same samples of Wiener process we have to store the state of the random number generator (NU in program (32)) and the Wiener process value $w(t_r)$ at each initial point t_r .

If $\varepsilon/10 < \delta_{r+1} < \varepsilon$, calculate the next step with the same step length $h_{r+1} = h_r$.

If $\delta_{r+1} < \varepsilon/10$, calculate the next step with double step length $h_{r+1} = 2h_r$.

The integration method can be both Runge-Kutta and Euler. But some examples show (see Section 6) that the Euler method with variable integration step length has no advantages compared to the Euler method with constant integration step length. In fact the R-K local truncation error decreases as $h \rightarrow 0$ faster than the Euler error. Therefore, if we can consider solution $x_{r+1}^{(2)}$ in expression (30) to be rather close to the exact one, it is not true for the Euler method. Hence we recommend to use the variable integration step method only for R-K schemes.

5. DESCRIPTION OF THE FORTRAN PROGRAM

The program package for simulating the n-dimensional stochastic differential equation (1) contains three standard subroutines STOCH, STH, STEP, the main program, and at least two user subroutines SIDE and OUTP which are called on by standard subroutines.

5.1 Main Program

The main program calls on the standard subroutine

```
CALL STOCH(XI,TMIN,TMAX,NX,NR,EPS,K,KMAX,KMIN,ISTEP,
METHOD,TPl,DP,NU,IS,W)
```

where the input arguments are:

- XI - vector of initial conditions of state variables
- TMIN - initial time of integration
- TMAX - upper limit of integration time
- NX - number of first-order equation n in (1)
- NR - number of input white noises m in (1)
- EPS - local truncation error in the variable integration step length method, ε
- K - defines the value of the integration step h (26) if ISTEP=0, otherwise K indicates the initial value of h
- KMAX - indicates the minimum value of the integration step length h_{\min} (25)
- KMIN - indicates the maximum value h_{\max}
- ISTEP - if ISTEP=0, the integration interval $[t_{\min}, t_{\max}]$ is divided into 2^K equal segments of length h (26), otherwise the variable integration step method is used
- METHOD - if METHOD=0, the subroutine obtains Itô solution by the Euler method, otherwise Stratonovich solution by the fourth-order Runge-Kutta method
- TPl - first point of time to output the solution by means of the user subroutine OUTP
- DP - increment of time to output the solution
- NU - initial state of the random number generator (odd number).

The argument IS is the output one which equals the number of integration steps. Vector W is the allocation vector.

In the main program the user has to define the input arguments and point out the real dimensions of vectors XI(NX) and W(7·NX+4·NR).

5.2 Standard Subroutines

Subroutine STOCH (Table 1) is the auxiliary subroutine for dynamic allocation of vectors in FORTRAN-4 language (see report [13]). This subroutine calls on the basic subroutine STH (Table 2) which organizes the numerical integration with a constant, if ISTEP=0, or with variable step length otherwise. The subroutine STH also contains the Wiener process generator (32) which calls on the library subroutine MCREDI.

Subroutine STH calls on the standard subroutine STEP (Table 3) to perform integration on one step by the Euler method (5'), if METHOD=0, or by R-K method (13) otherwise.

5.3 User Subroutines

5.3.1 Subroutine SIDE. Subroutine STEP calls on the user subroutine

SIDE(T+H,H,XH,R,DX)

where the input arguments are

- T - current time, t_r
- H - current integration step length, h_r , when using constant step technique, $h_r/2$ otherwise
- X - initial vector $\vec{x}(t_r)$ on elementary integration interval (t_r, t_{r+1})
- R - vector of Wiener process increments $\Delta\vec{w}_r$.

The user has to write the program to calculate the output vector argument DX of dimension NX in such a way.

For the Euler method (5') DX is the vector state variables increments, for the R-K method (13) it is the vector coefficient \vec{K}_ℓ ($\ell=1,2,3,4$).

Vector DX contains coefficients of the stochastic differential equation (1). Let the $n \times 1$ vector $A(I) = \alpha_i(\vec{x}(t_r), t_r)$. Note that in many practical cases matrix $\| b_{ij} \|$ in equation (1) contains a lot of zero elements. Therefore the user has to write down the $n \times m$ matrix $\| b_{ij} \|$ as a vector $B(J) = b_{ij}(\vec{x}_r, t_r)$ for non-zero elements b_{ij} . The dimension of B is equal to data of non-zero elements in matrix $\| b_{ij} \|$.

In the FORTRAN notation the i :th component of vector DX is equal to

$$DX(I) = A(I) * H + B(\dots) * R(1) + \dots + B(\dots) * R(NR), \quad (35)$$

$$I = 1, 2, \dots, NX.$$

Thus, in subroutine SIDE, the user has to define the vector coefficients A(I) and B(J) and write down expression (35) where H and R(J) are input arguments defined into basic subroutine STEP.

Due to allocation vector techniques the dimension of the vectors X, R, and DX can be described like this:

```
DIMENSION X(1),DX(1),R(1)
```

For vectors A and B the user has to point out their real dimensions.

5.3.2 Subroutine OUTP. Basic subroutine STH calls on the subroutine

```
OUTP(T,X,WIENER)
```

to output the current time T, the vector state variables X and the Wiener process WIENER.

All the arguments of subroutine OOTP are the input arguments defined into the basic subroutine STH.

Dimensions of the vector X and WIENER are described in such a manner:

```
DIMENSION X(1),WIENER(1).
```

TABLE 1

```

001 C NAME: STOCH
002 C -----
003 C SUBTITLE: SOLUTION OF THE NONLINEAR ITO OR STRATONOVICH STOCHASTIC
004 C DIFFERENTIAL EQUATIONS
005 C
006 C KEYWORDS:
007 C -----
008 C SOLUTION, SYSTEM OF NONLINEAR STOCHASTIC DIFFERENTIAL EQUATIONS,
009 C ITO EQUATION, STRATONOVICH EQUATION
010 C
011 C IMPLEMENTOR: VSEVOLOD D. RAZEVIĆ DATE: 1977-05-19
012 C -----
013 C
014 C INSTITUTE:
015 C -----
016 C DEPARTMENT OF AUTOMATIC CONTROL
017 C LUND INSTITUTE OF TECHNOLOGY, SWEDEN
018 C
019 C ACCEPTED: VERSION: 1
020 C -----
021 C
022 C =====
023 C
024 C PURPOSE
025 C =====
026 C
027 C ITO OR STRATONOVICH SOLUTION OF THE SYSTEM OF N FIRST-ORDER NONLINEAR
028 C STOCHASTIC DIFFERENTIAL EQUATIONS  $X' = A(X, T) + B(X, T) * W'$  WITH
029 C CONSTANT OR VARIABLE LENGTH STEPS
030 C
031 C USAGE
032 C =====
033 C
034 C PROGRAM TYPE: SUBROUTINE
035 C -----
036 C
037 C ARGUMENTS:
038 C -----
039 C STOCH(XI, TMIN, TMAX, NX, NR, EPS, K, KMAX, KMIN, ISTEP, METHOD,
040 C 1TP1, DP, NU, IS, W)
041 C
042 C XI -- VECTOR OF INITIAL CONDITIONS OF STATE VARIABLES
043 C TMIN -- INITIAL TIME OF INTEGRATION
044 C TMAX -- UPPER LIMIT OF INTEGRATION TIME
045 C NX -- NUMBER OF EQUATIONS
046 C NR -- NUMBER OF INPUT WHITE NOISES
047 C EPS -- LOCAL TRUNCATION ERROR
048 C K -- INDICATE THE INITIAL VALUE OF THE INTEGRATION STEP
049 C H=(TMAX-TMIN)/2**K
050 C KMAX -- INDICATE THE MINIMUM VALUE OF THE INTEGRATION STEP
051 C HMIN=(TMAX-TMIN)/2**KMAX
052 C KMIN -- INDICATE THE MAXIMUM VALUE OF THE INTEGRATION STEP
053 C HMAX=(TMAX-TMIN)/2**KMIN
054 C ISTEP -- IF ISTEP=0, INTEGRATION INTERVAL DIVIDED INTO 2**K EQUAL
055 C SEGMENTS OF LENGTH H, OTHERWISE ARE USED THE VARIABLE
056 C INTEGRATION STEP
057 C METHOD-- IF METHOD=0, SUBROUTINE OBTAINS ITO SOLUTION BY EULER METHOD,
058 C OTHERWISE STRATONOVICH SOLUTION BY THE FOURTH-ORDER
059 C RUNGE-KUTTA METHOD
060 C TP1 -- FIRST POINT OF TIME TO OUTPUT THE SOLUTION BY MEANS USER
061 C SUBROUTINE OUTP
062 C DP -- INCREMENT OF TIME TO OUTPUT THE SOLUTION
063 C NU -- INITIAL STATE OF RANDOM NUMBER GENERATOR

```

TABLE 1 (continued)

```

064 C      IS - NUMBER OF THE INTEGRATION STEPS
065 C      W  - ALLOCATION VECTOR HAVING DIMENSION 7*NX+4*NR
066 C
067 C      NOTES:
068 C      -----
069 C      1) USER COMPOSES THE MAIN PROGRAM WHICH CALLS ON SUBROUTINE STOCH
070 C      2) USER COMPOSES SUBROUTINE SIDE(T,H,X,R,DX)
071 C
072 C      INPUT ARGUMENTS:
073 C      -----
074 C      T  - CURRENT TIME
075 C      H  - CURRENT INTEGRATION STEP LENGTH
076 C      X  - CURRENT STATE VARIABLES VECTOR
077 C      R  - WIENER PROCESS INCREMENT VECTOR
078 C
079 C      OUTPUT ARGUMENT
080 C      -----
081 C      DX - UNIT INCREMENT VECTOR EQUALS TO DX(I)=A(X,T)*H+B(X,T)*R(J)
082 C          (I=1,2,...,NX, J=1,2,...,NR). NONLINEAR VECTOR COEFFICIENTS
083 C          A(X,T) AND B(X,T) ARE DESCRIBED BY USER, DIMENSION
084 C          OF A(X,T) EQUALS TO NX, MAXIMUM DIMENSION OF B(X,T)
085 C          EQUALS TO NX*NR.
086 C      3) USER COMPOSES SUBROUTINE OUTP(T,X,WIENER) TO OUTPUT
087 C          THE SOLUTION X(I) AND WIENER PROCESS WIENER(J).
088 C      4) SUBROUTINE STOCH CALLS ON INNER SUBROUTINES STH AND STEP
089 C      5) SUBROUTINE STH CALLS ON LIBRARY SUBROUTINE MCREDI (RECTANGULAR
090 C          RANDOM NUMBER GENERATOR)
091 C
092 C      METHOD
093 C      =====
094 C
095 C      IN STH IS USED EULER OR FORTH-ORDER RUNGE-KUTTA METHOD BOTH WITH
096 C      CONSTANT AND VARIABLE INTEGRATION STEP LENGTH . THE CURRENT STEP
097 C      IS HALFING AND DOUBING IN SUCH A MANNER TO GET THE SAME SAMPLES
098 C      OF NOISE WHILE SOLVING THE EQUATION WITH DIFFERENT VALUES OF
099 C      EPS OR K.
100 C
101 C      REFERENCES:
102 C      -----
103 C      1. D.J.WRIGHT, IEEE TRANS. ON AUTOMATIC CONTROL, V.AC-19, N 1, 1974
104 C      2. N.NIKITIN, S.PERVACHEV, V.RAZEVIK, AUTOMATION AND
105 C          REMOTE CONTROL, N 4, 1975.
106 C      3. C.W.GEAR, NUMERICAL INITIAL VALUE PROBLEMS IN ORDINARY
107 C          DIFFERENTIAL EQUATIONS, 1971.
108 C
109 C      CHARACTERISTICS
110 C      =====
111 C
112 C      REVISIONS:
113 C      -----
114 C
115 C      -----
116 C
117 C      SUBROUTINE STOCH(XI,TMIN,TMAX,NX,NR,EPS,K,KMAX,KMIN,ISTEP,METHOD,
118 C      1TP1,DP,NU,IS,W)
119 C
120 C      DIMENSION XI(1),W(1)
121 C
122 C
123 C      KW1=1
124 C      KW2=KW1+NX
125 C      KW3=KW2+NX
126 C      KW4=KW3+NX
127 C      KW5=KW4+NR

```

L28 KW6=KW5+NR
L29 KW7=KW6+NR
L30 KW8=KW7+NR
L31 KW9=KW8+NX
L32 KW10=KW9+NX
L33 KW11=KW10+NX

C
C
C

CALL ON THE BASIC SUBROUTINE STH

L37 CALL STH(XI,TMIN,TMAX,NX,NR,EPS,K,KMAX,KMIN,ISTEP,METHOD,
L38 1TP1,DP,NU,IS,W(KW1),W(KW2),W(KW3),W(KW4),W(KW5),W(KW6),W(KW7),
L39 2W(KW8),W(KW9),W(KW10),W(KW11))
L40 RETURN
L41 END

TABLE 2

```

01 C
02 C THIS IS BASIC SUBROUTINE TO SOLVE STOCHASTIC DIFFERENTIAL EQUATION
03 C
04 C SUBROUTINE STH(X,TMIN,TMAX,NX,NR,EPS,K,KMAX,KMIN,ISTEP,METHOD,
05 C 1TP1,DP,NU,IS,X1,X2,XH2,WIENER,WIENE1,R,R1,S1,S2,S3,S4)
06 C
07 C DIMENSION X(1),X1(1),X2(1),XH2(1),WIENER(1),WIENE1(1),R(1),R1(1),
08 C 1S1(1),S2(1),S3(1),S4(1)
09 C
10 C DATA PI2/6.283185307/
11 C
12 C DETERMINE ZERO INITIAL CONDITIONS FOR WIENER PROCESSES
13 C
14 C DO 1 J=1,NR
15 C 1 WIENER(J)=0.0
16 C
17 C ... FOR TIME T, SWITCH IS AND VARIABLE TPR RUNNIG
18 C THE OUTPUTING SUBROUTINE OUTP
19 C
20 C T=TMIN
21 C IS=0
22 C TPR=TP1
23 C EPS1=EPS/10.0
24 C
25 C DETERMINE THE MINIMUM AND MAXIMUM VALUES OF INTEGRATION
26 C STEP LENGTH
27 C
28 C HMIN=(TMAX-TMIN)/2**KMAX
29 C HMAX=(TMAX-TMIN)/2**KMIN
30 C SHM=SQRT(HMIN)
31 C
32 C CALL ON USER SUBROUTINE OUTP TO OUTPUT THE INITIAL CONDITIONS
33 C
34 C CALL OUTP(T,X,WIENER)
35 C
36 C THE BEGINING OF THE INTEGRATION,
37 C DETERMINE THE CURRENT INTEGRATION LENGTH STEP
38 C
39 C 2 H=(TMAX-TMIN)/2**K
40 C H2=H+H
41 C
42 C STORAGE THE PRECEDING VALUE OF STATE NU OF RANDOM NUMBER GENERATOR
43 C AND WIENER PROCESS FOR A RESTART
44 C
45 C N1=NU
46 C DO 3 J=1,NR
47 C WIENE1(J)=WIENER(J)
48 C 3 R(J)=0.0
49 C
50 C CALCULATE THE NUMBER OF CONSECUTIVE SAMPLINGS KR
51 C
52 C KR=INT(H/HMIN+0.1)
53 C
54 C GENERATE GAUSSIAN INCREMENTS R(J) OF WIENER PROCESS AND
55 C THE WIENER PROCESS AT THE POINT T+H
56 C
57 C DO 4 I=1,KR
58 C DO 4 J=1,NR
59 C 31 CALL MCREDI(NU,REC1)
60 C CALL MCREDI(NU,REC2)
61 C IF(REC1,EQ,0.0)GOTO 31
62 C GS=SQRT(-2.0*ALOG(REC1))*COS(PI2*REC2)
63 C 4 R(J)=R(J)+GS

```

TABLE 2 (continued)

```

064 DO 5 J=1,NR
065 R(J)=R(J)*SHM
066 5 WIENER(J)=WIENER(J)+R(J)
067 C
068 CALL ON SUBROUTINE STEP TO CALCULATE THE STATE VARIABLES
069 X1(I) ON THE FIRST STEP LENGTH H
070 C
071 CALL STEP(T,H,X,R,X1,NX,NR,METHOD,S1,S2,S3,S4)
072 C
073 IF ISTEP=0, PUT THE VALUES X1(I) TO THE CURRENT STATE VARIABLES
074 VECTOR X(I) AND GO TO THE BOTTOM OF SUBROUTINE
075 C
076 IF(ISTEP,NE.0)GOTO 7
077 T=T+H
078 DO 6 I=1,NX
079 6 X(I)=X1(I)
080 IS=IS+1
081 GOTO 16
082 C
083 IF ISTEP NOT EQUALS TO ZERO, COMPUTE THE NEXT STEP
084 OF THE LENGTH H AND THE LARGER STEP OF THE LENGTH 2*H
085 C
086 GENERATE GAUSSIAN INCREMENTS OF WIENER PROCESS ON THE NEXT STEP R1(J)
087 AND CALCULATE WIENER PROCESS AT THE POINT T+2*H
088 C
089 7 DO 8 J=1,NR
090 8 R1(J)=0.0
091 DO 10 I=1,KR
092 DO 10 J=1,NR
093 9 CALL MCREDI(NU,REC1)
094 CALL MCREDI(NU,REC2)
095 IF(REC1,EQ,0.0)GOTO 9
096 GS=SQRT(-2.0*ALOG(REC1))*COS(PI2*REC2)
097 10 R1(J)=R1(J)+GS
098 DO 11 J=1,NR
099 R1(J)=R1(J)*SHM
100 WIENER(J)=WIENER(J)+R1(J)
101 11 R(J)=R(J)+R1(J)
102 C
103 CALL ON SUBROUTINE STEP TO CALCULATE XH2(I) ON THE LARGE STEP 2*H
104 C
105 CALL STEP(T,H2,X,R,XH2,NX,NR,METHOD,S1,S2,S3,S4)
106 C
107 CALL ON SUBROUTINE STEP TO CALCULATE X2(I) ON THE SECOND STEP H
108 C
109 CALL STEP(T+H,H,X1,R1,X2,NX,NR,METHOD,S1,S2,S3,S4)
110 C
111 CALCULATE THE LARGEST ERROR DM
112 C
113 DM=0.0
114 DO 12 I=1,NX
115 DE=ABS(XH2(I)-X2(I))
116 XA=ABS(X2(I))
117 IF(XA.GT.1.0)DE=DE/XA
118 IF(DE.GT.DM)DM=DE
119 12 CONTINUE
120 C
121 IS DM LARGER EPS?
122 C
123 IF(DM-EPS)13,13,20
124 C
125 IF DM<EPS, DETERMINE CURRENT TIME T=T+H AND CURRENT STATE VARIABLES
126 X(I)=X2(I). IF IN ADDITION DM<EPS/10 DOUBL THE LENGTH STEP
127 C

```

TABLE 2 (continued)

```

28 13 IF(DM.LT.EPS1)K=K-1
29   IF(K.LT.KMIN)K=KMIN
30 14 T=T+H2
31   IS=IS+2
32   DO 15 I=1,NX
33 15 X(I)=X2(I)
34  C
35  C   CHECK THE NECESSERITY TO CALL ON SUBROUTINE OUTP
36  C
37 16 IF(T-TPR)2,19,19
38 19 TPR=TPR+DP
39   CALL OUTP(T,X,WIENER)
40  C
41  C   IS T LARGER THAN TMAX?
42  C
43   IF(T+HMIN/2.0.LT.TMAX)GOTO 2
44   GOTO 25
45  C
46  C   IF DM>EPS, REPEAT THE CALCULATIONS HALFING THE LENGTH STEP
47  C
48 20 K=K+1
49  C
50  C   IS K LARGER THAN KMAX?
51  C
52   IF(KMAX-K)21,22,22
53  C
54  C   IF K>KMAX, ELIMINATE K AND GO TO THE NEXT STEP
55  C
56 21 K=KMAX
57   GOTO 14
58  C
59  C   IF K LESS OR EQUAL KMAX RENEW THE STATE OF THE RANDOM NUMBER
60  C   GENERATOR AND WIENER PROCESSES AND REPEAT THE CALCULATIONS
61  C
62 22 NU=N1
63   DO 23 J=1,NR
64 23 WIENER(J)=WIENE1(J)
65   GOTO 2
66 25 RETURN
67   END

```


TABLE 3

```

001 C
002 C THIS IS INNER SUBROUTINE TO PERFORM INTEGRATION ON ONE STEP,
003 C IF METHOD=0, SUBROUTINE STEP OBTAINS ITO SOLUTION BY EULER
004 C METHOD, OTHERWISE STRATONOVICH SOLUTION BY THE FORTH-ORDER
005 C RUNGE-KUTTA METHOD.
006 C
007 C SUBROUTINE STEP(T,H,X,R,XH,NX,NR,METHOD,S1,S2,S3,S4)
008 C
009 C DIMENSION X(1),R(1),XH(1),S1(1),S2(1),S3(1),S4(1)
010 C
011 C CALL ON USER SUBROUTINE SIDE TO DETERMINE THE INCREMENT OF
012 C STATE VARIABLES ON THE STEP LENGTH H WITH INITIAL
013 C CONDITIONS X(I) AT THE TIME T.
014 C
015 C CALL SIDE(T,H,X,R,S1)
016 C
017 C IS METHOD=0?
018 C
019 C IF(METHOD.NE.0)GOTO 2
020 C
021 C IF METHOD=0 CALCULATE THE NEXT SAMPLE XH(I) OF EULER SOLUTION
022 C AT THE POIN T+H AND GO TO THE BOTTOM OF THE SUBROUTINE
023 C
024 C DO 1 I=1,NX
025 1 XH(I)=S1(I)+X(I)
026 GOTO 7
027 C
028 C IF METHOD NOT EQUAL TO ZERO CALCULATE THE NEXT RUNGE-KUTTA
029 C COEFFICIENTS
030 C
031 C 2 DO 3 I=1,NX
032 3 XH(I)=X(I)+S1(I)*0.5
033 CALL SIDE(T+H*0.5,H,XH,R,S2)
034 DO 4 I=1,NX
035 4 XH(I)=X(I)+S2(I)*0.5
036 CALL SIDE(T+H*0.5,H,XH,R,S3)
037 DO 5 I=1,NX
038 5 XH(I)=X(I)+S3(I)
039 CALL SIDE(T+H,H,XH,R,S4)
040 C
041 C DETERMINE THE NEXT SAMPLE XH(I) OF STRATONOVICH SOLUTION
042 C
043 C DO 6 I=1,NX
044 6 XH(I)=(S1(I)+2.0*S2(I)+2.0*S3(I)+S4(I))/6.0+X(I)
045 7 RETURN
046 END

```

6. EXAMPLES

Example 1. Consider the $I\hat{t}^0$ solution of equation

$$dx(t) = cx dt + gx dw(t) \quad (36)$$

by the Euler method (5') with constant integration step length on time interval $t \in [0,1]$ with initial condition $x(0) = 1$.

Choose the number of samples $x(t)$ to be printed equal to 100 ($TP1=DP=1/100$), the integration step size $h = 1/2^{10}$ ($K=10$), $h_{\min} = 1/2^{15}$ ($KMAX=15$).

The number of equations in (36) equals $NX=1$ and the number of white noises $NR=1$. Therefore the dimension of the allocation vector W equals $7*NX+1*NR=11$. Choose the initial value of random number generator $NU = 65317$.

The main program is shown in Table 4. Note that variables $KMIN$ and EPS are not used in the constant integration step length method.

The coefficients of equation (36) and solution increment (35) are described at the user subroutine $SIDE$, Table 5, where coefficients $c = -1$, $g = 1$.

The user subroutine $OUTP$ (Table 6) outputs the current time T , the solution $X(1)$ and the Wiener process $WIENER(1)$ on a line-printer.

Besides, at Fig. 4 we compare the rate of convergence of several $I\hat{t}^0$ integration procedures:

- 1) Euler method (5')
- 2) modified Euler method (14)
- 3) second approximation (15)

TABLE 4

```

001      C   PROGRAM MAIN
002          DIMENSION XI(1),W(11)
003          COMMON /OUT/ISTEP,METHOD
004          TMIN=0.0
005          TMAX=1.0
006          DP=(TMAX-TMIN)/100.0
007          TP1=DP
008          KMAX=15
009          K=10
010          ISTEP=0
011          METHOD=0
012          XI(1)=1.0
013          NX=1
014          NR=1
015          NU=65317
016          WRITE(6,1)TMIN,TMAX,EPS,TP1,DP,KMAX,KMIN,K,NX,NR,ISTEP,METHOD
017      1   FORMAT(5F11.8,7I8)
018          CALL STOCH(XI,TMIN,TMAX,NX,NR,EPS,K,KMAX,KMIN,ISTEP,METHOD,
019      1TP1,DP,NU,IS,W)
020          WRITE(6,100)IS
021      100  FORMAT(I10)
022          STOP
023          END

```

TABLE 5

```

001          SUBROUTINE SIDE(T,H,X,R,DX)
002          DIMENSION X(1),DX(1),R(1)
003          DIMENSION A(1),B(1)
004          DATA C,G/-1.0,1.0/
005          A(1)=C*X(1)
006          B(1)=G*X(1)
007          DX(1)=A(1)*H+B(1)*R(1)
008          RETURN
009          END

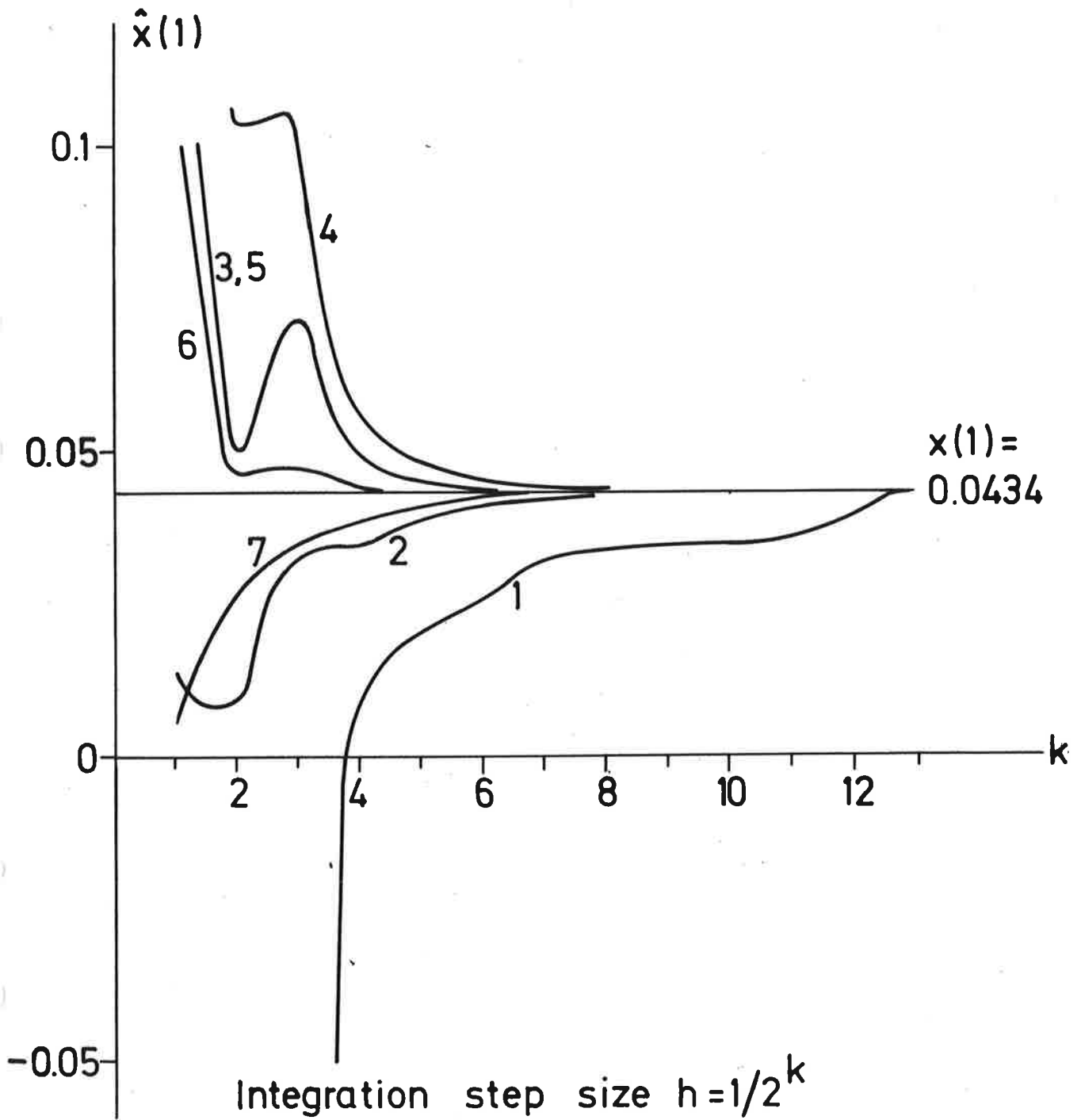
```

TABLE 6

```

001          SUBROUTINE OUTP(T,X,WIENER)
002          DIMENSION X(1),WIENER(1)
003          WRITE(6,10)T,X(1),WIENER(1)
004      10  FORMAT(3F15.8)
005          RETURN
006          END

```



(a)

Fig.4a. Equation (36)

- 1 - Euler method (5')
- 2 - modified Euler method (14)
- 3 - second approximation (15)
- 4 - standard Euler-Cauchy method applied to equation (17)
- 5 - Euler-Cauchy method analogues to (16)
- 6 - standard Runge-Kutta method applied to equation (17)
- 7 - Runge-Kutta method (16)

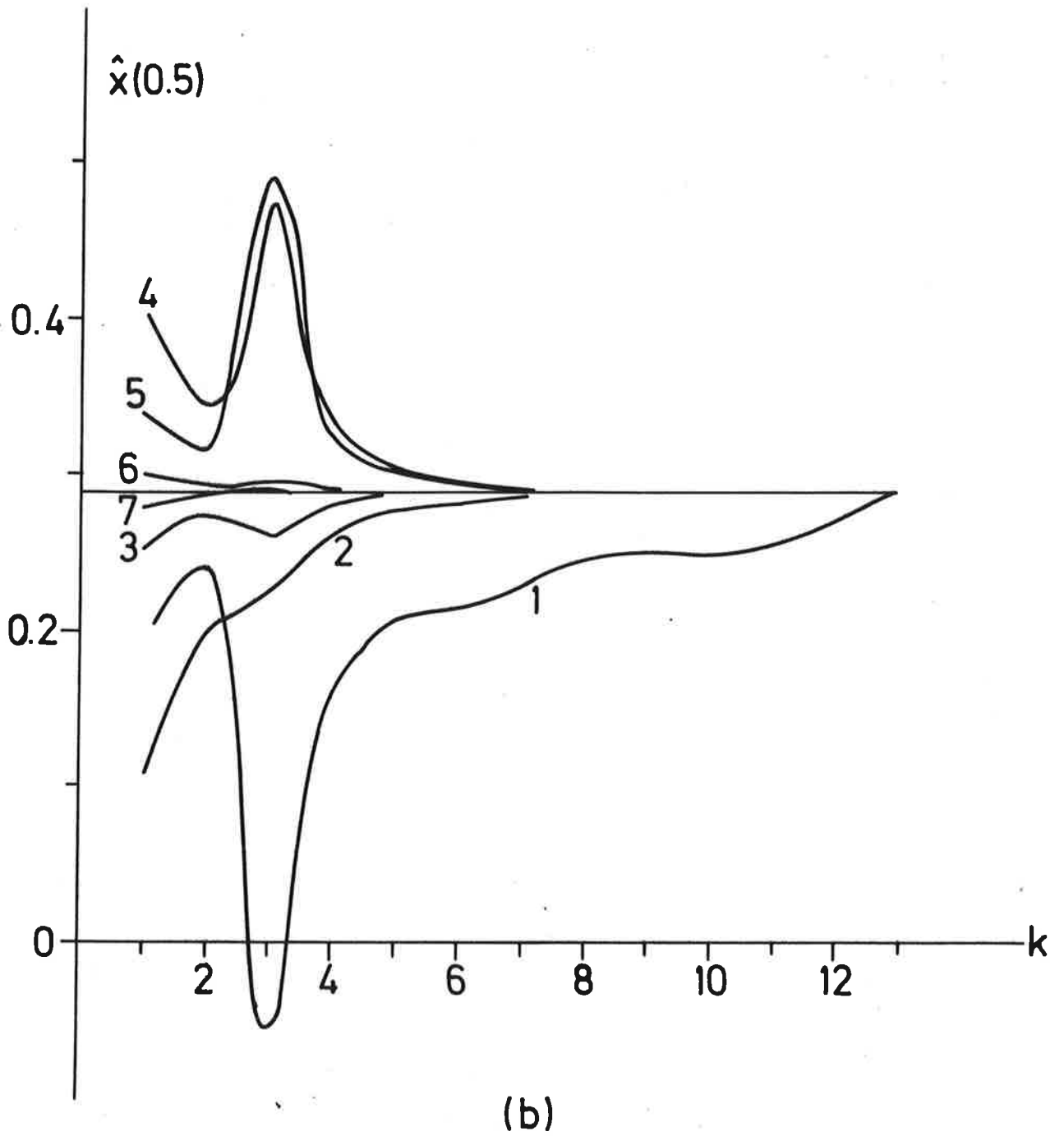


Fig.4b. Equation $dx = -\sin(x) + \sqrt{|x|} dw(t)$

- 4) standard Euler-Cauchy method (second-order R-K method) applied to equivalent Stratonovich equation (17)
- 5) Euler-Cauchy method analogues to (16)
- 6) standard Runge-Kutta method (18) applied to equivalent Stratonovich equation (17)
- 7) Runge-Kutta method (16).

Fig. 4a,b shows that the relative rates of convergence were as follows:

- quickest - Runge-Kutta methods (16), (18)
 - second approximation (15)
- intermediate - Euler-Cauchy methods
 - modified Euler method (14)
- slowest - Euler method (5').

Example 2. Let's find the Itô solution of equation (36) by the step variable fourth-order Runge-Kutta method (18) applied to equivalent Stratonovich equation (17)

$$dx(t) = (c - \frac{1}{2} g^2) x dt + gx dw(t). \quad (37)$$

The new main program and the subroutine SIDE are shown in Tables 7 and 8 accordingly. Subroutine OUTP is the same as in Example 1 (see Table 6).

Equation (36) has the analytical Itô solution

$$x(t) = x(0) \exp \left\{ (c - \frac{1}{2} g^2) t + gw(t) \right\}. \quad (38)$$

Hence it is possible to find the accuracy of numerical solution

$$\delta(t) = \left| \frac{x(t) - \hat{x}(t)}{x(t)} \right| \quad (39)$$

where $x(t)$ is the exact solution (38), $\hat{x}(t)$ is the numerical solution.

TABLE 7

```

001      C   PROGRAM MAIN
002          DIMENSION XI(1),W(11)
003          COMMON /OUT/ISTEP,METHOD
004          TMIN=0.0
005          TMAX=1.0
006          DP=(TMAX-TMIN)/100.0
007          TP1=DP
008          KMAX=15
009          K=10
010          KMIN=4
011          EPS=1.0E-06
012          ISTEP=1
013          METHOD=1
014          XI(1)=1.0
015          NX=1
016          NR=1
017          NU=65317
018          WRITE(6,1)TMIN,TMAX,EPS,TP1,DP,KMAX,KMIN,K,NX,NR,ISTEP,METHOD
019      1   FORMAT(5F11.8,7I8)
020          CALL STOCH(XI,TMIN,TMAX,NX,NR,EPS,K,KMAX,KMIN,ISTEP,METHOD,
021      1TP1,DP,NU,IS,W)
022          WRITE(6,100)IS
023      100  FORMAT(I10)
024          STOP
025          END

```

TABLE 8

```

001          SUBROUTINE SIDE(T,H,X,R,DX)
002          DIMENSION X(1),DX(1),R(1)
003          DIMENSION A(1),B(1)
004          DATA C,G/-1.0,1.0/
005          A(1)=C*X(1)
006          B(1)=G*X(1)
007          DX(1)=(A(1)-0.5*G*G*X(1))*H+B(1)*R(1)
008          RETURN
009          END

```

In order to compare the accuracy of Euler and R-K schemes with constant and variable step lengths, Fig. 5 shows the accuracy δ (39) at time $t = 1$ as a function of number of steps. We can conclude that the R-K method with variable integration step length is the most available integration method for stochastic differential equations.

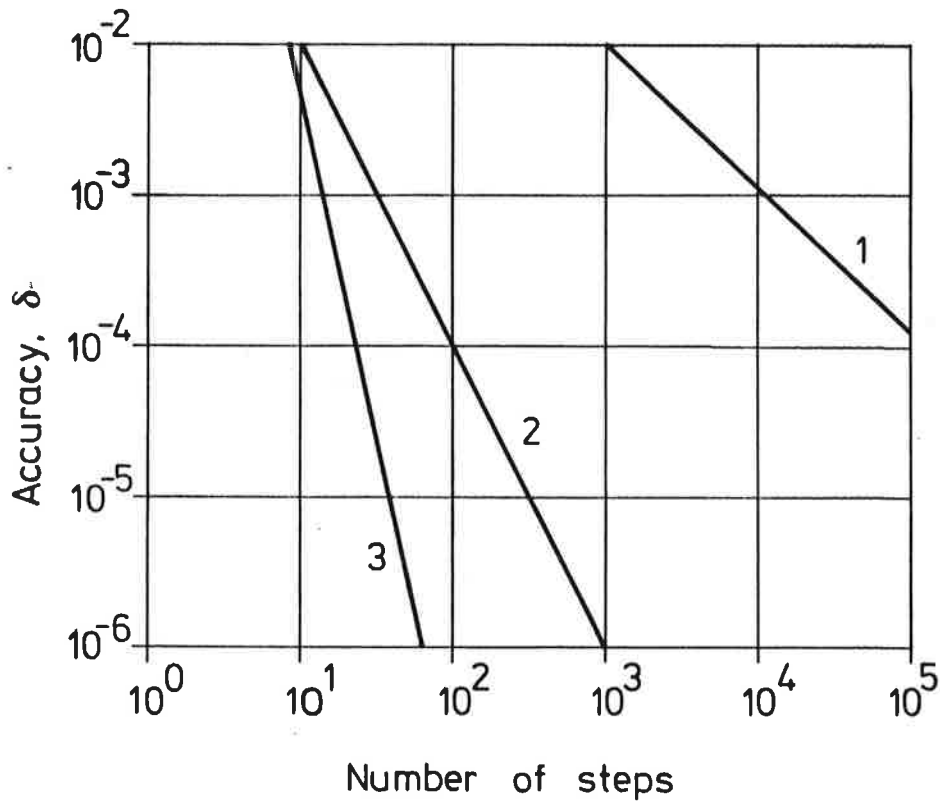


Fig. 5 - Accuracy of methods with constant and variable step size.

- 1 - Euler method with constant step length.
- 2 - Runge-Kutta method with constant step length.
- 3 - Runge-Kutta method with variable step length.

Example 3. Consider the system of two first-order stochastic differential equations

$$\begin{cases} \frac{dx_1}{dt} = x_2 + b_1 \xi_1(t) \\ \frac{dx_2}{dt} = v - x_2 - a x_1 e^{-x_1^2/L} + b_2(x_2) \xi_2(t) \end{cases} \quad (40)$$

where

$$b_2(x_2) = \frac{g \cdot a}{\tau} \frac{1+10 x_2^2}{5+10 x_2^2}.$$

The user routines to obtain Itô solution of (40) by constant step Euler method and to display results with IDPAC system are shown in Tables 9-11. The dialog on teletype is shown in Table 12. The displayed phase plane of system (40) is shown in Fig. 6.

Example 4. In article [8] second-order phase-lock loop differential equation was studied and it was discovered in practice that "the Euler integration is greatly superior to Runge-Kutta methods from a convergence-rate accuracy criterion". It is difficult to comment on such a statement as no mathematical expressions to the numerical schemes used was pointed out.

We consider that the Runge-Kutta methods (13), (16), (18) (and their multidimensional analogues) have better accuracy than the Euler method (5') or have, in some special examples [4, p.186], at least the same accuracy.

That's why we implement our computer program to simulate the phase-lock loop [8]:

$$\begin{cases} \frac{dx_1}{dt} = x_2, \\ \frac{dx_2}{dt} = -\sin(x_1) - \cos(x_1) \xi_1(t) - \sin(x_1) \xi_2(t) \end{cases} \quad (41)$$

TABLE 9

```

001      C      PROGRAM MAIN
002          DIMENSION XI(2),W(22)
003          COMMON /PLOT/IP
004          TMIN=0.0
005          TMAX=1.0
006          NPR=1000
007          IP=NPR+1
008          DP=(TMAX-TMIN)/NPR
009          TP1=DP
010          KMAX=12
011          K=12
012          ISTEP=0
013          METHOD=0
014          XI(1)=-2.0
015          XI(2)=1.25
016          NX=2
017          NR=2
018          NU=65317
019          CALL OUTP(T,XI,W)
020          IP=0
021          WRITE(6,1)TMIN,TMAX,EPS,TP1,DP,KMAX,KMIN,K,NX,NR,ISTEP,METHOD
022      1  FORMAT(5F11.8,7I8)
023          CALL STOCH(XI,TMIN,TMAX,NX,NR,EPS,K,KMAX,KMIN,ISTEP,METHOD,
024      1TP1,DP,NU,IS,W)
025          WRITE(6,100)IS
026      100  FORMAT(I10)
027          IP=-1
028          CALL OUTP(T,XI,W)
029          STOP
030          END

```

TABLE 10

```

001      SUBROUTINE SIDE(T,H,X,R,DX)
002          DIMENSION X(1),DX(1),R(1)
003          DIMENSION A(2),B(2)
004          DATA G,V,AK,TAY,RB/0.3,1.25,5.0,0.5,0.4/
005          B(2)=BB
006          XX=X(1)*X(1)
007          X10=10.0*X(2)*X(2)
008          A(1)=X(2)
009          A(2)=(V-X(2)-AK*X(1)*EXP(-XX))/TAY
010          B(1)=G*AK/TAY*(1.0+X10)/(5.0+X10)
011          DX(1)=A(1)*H+B(2)*R(2)
012          DX(2)=A(2)*H+B(1)*R(1)
013          RETURN
014          END

```

TABLE 11

```

001      SUBROUTINE  OUTP(T,X,WIENER)
002      DIMENSION X(1),WIENER(1),IT(10),FILN(2)
003      COMMON /PLOT/IP
004      DATA FILN(2)/4H BIN/
005      IF(IP)300,200,100
006      100  IT(1)=IP
007          IT(2)=3
008          IT(4)=0
009          IT(9)=0
010          WRITE(9,1000)
011          READ(8,1001)FILN(1)
012          CALL ENTER(1,FILN)
013          WRITE(1)IT
014          RETURN
015      1000 FORMAT(' ENTER FILE NAME')
016      1001 FORMAT(A5)
017      200  WRITE(1)T,X(1),X(2)
018          RETURN
019      300  CALL CLOSE(1)
020          RETURN
021          END

```

TABLE 12

\$FOR

FPF4X V3A000
>BL←MAIN

END PASS1
FPF4X V3A000
>BL←SIDE

DOS-15 UV3A000
\$A RK <NEW> -1

\$GLOAD

LOADER V3A000
>←MAIN,SIDE,OUTP,STOCH,STH,STEP

ENTER FILE NAME
DATA

STOP 000000

DOS-15 YV3A999
\$A RK <PAC> -4/RK 3,4,5,7,15,16

\$BUFFS 5

\$E IDPAC

IDPAC V2D

>
PLOT 4 DATA(2)←DATA(3) -2 2

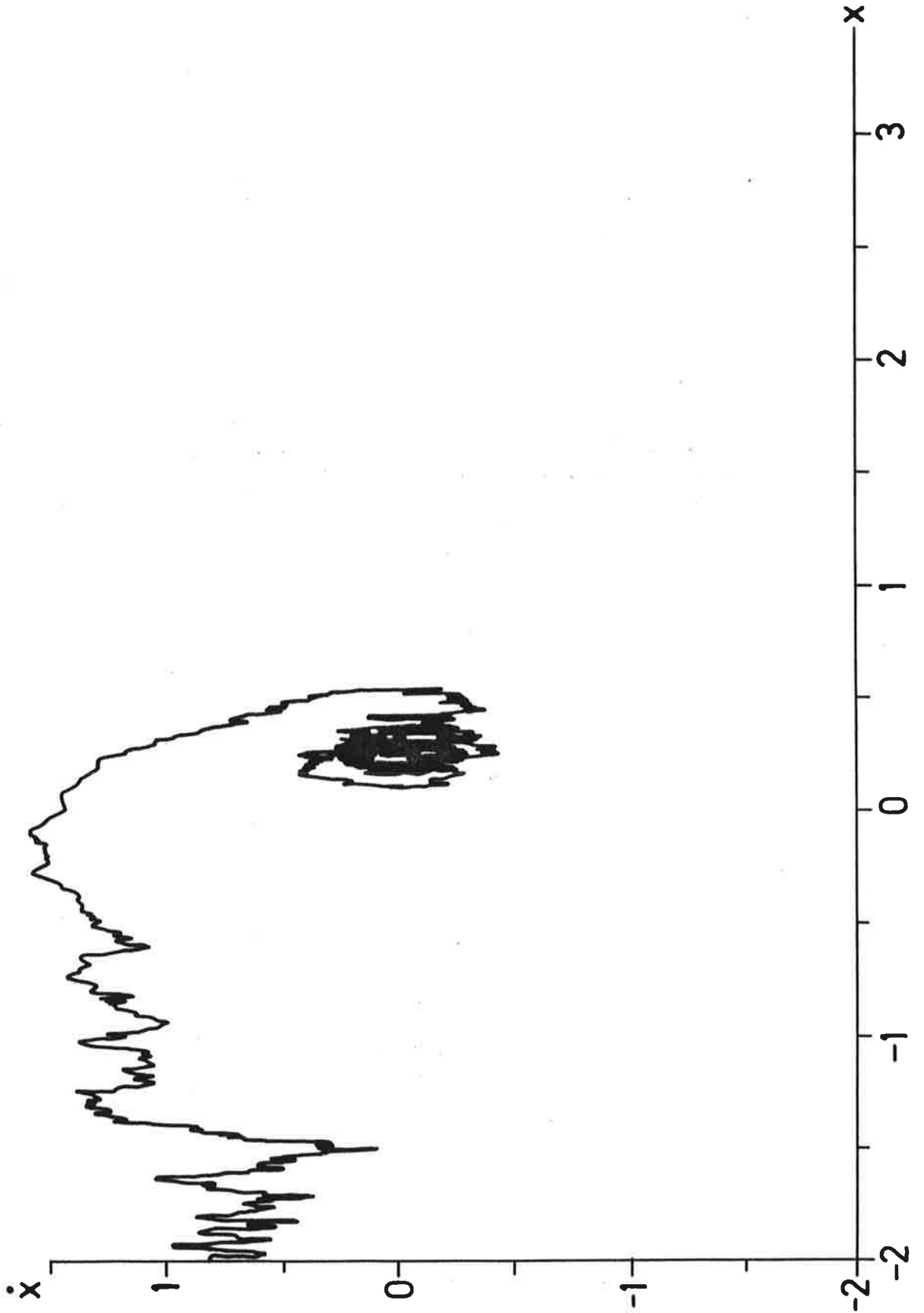


Fig.6. Phase plane

where $\xi_1(t)$ and $\xi_2(t)$ are uncorrelated white random processes with unite spectral density.

In [8] is shown that equation (41) has the same Itô and Stratonovich solutions as the term (20) is equal to zero. So, we solve (41) with Euler (5') and Runge-Kutta (13) methods. Listings of user subroutines are shown in Tables 13-15. Results of simulation during time $t \in [0,1]$ and initial conditions $x_1(0) = \pi/4$, $x_2(0) = \pi/4$ with different integration step size $h = 1/2^K$, $K = 1,2,\dots,14$, are shown in Tables 16-17. The solutions $x_1(1)$, $x_2(1)$ as functions of parameter K are shown in Fig. 7 which proves that the Runge-Kutta method (13) has a higher rate of convergence than the Euler method (5').

TABLE 13

```

001      C      PROGRAM MAIN
002          DIMENSION XI(2),W(22)
003          TMIN=0.0
004          TMAX=1.0
005          DP=TMAX
006          TP1=DP
007          KMAX=14
008          ISTEP=0
009          METHOD=1
010          NX=2
011          NR=2
012          WRITE(6,33)
013      33  FORMAT(5X,19H RUNGE-KUTTA METHOD)
014          DO 1 K=1,KMAX
015          WRITE(6,22)K
016      22  FORMAT(I6)
017          NU=65317
018          XI(1)=0.785
019          XI(2)=0.785
020          CALL STOCH(XI,TMIN,TMAX,NX,NR,EPS,K,KMAX,KMIN,ISTEP,METHOD,
021      1  TP1,DP,NU,IS,W)
022      1  CONTINUE
023          STOP
024          END

```

TABLE 14

```

001  SUBROUTINE SIDE(T,H,X,R,DX)
002  DIMENSION X(1),DX(1),R(1)
003  DIMENSION A(2)
004  S=SIN(X(1))
005  C=COS(X(1))
006  A(1)=X(2)
007  A(2)=-S
008  B=C*R(1)+S*R(2)
009  DX(1)=A(1)*H
010  DX(2)=A(2)*H-B
011  RETURN
012  END

```

TABLE 15

```

001  SUBROUTINE OUTP(T,X,WIENER)
002  DIMENSION X(1),WIENER(1)
003  WRITE(6,1)T,X(1),X(2)
004      1  FORMAT(3F18.10)
005  RETURN
006  END

```

TABLE 16

RUNGE-KUTTA METHOD			
K=1	t	x_1	x_2
	0.0000000000	0.7849999964	0.7849999964
	1.0000000000	0.7559317946	0.0567011535
2	0.0000000000	0.7849999964	0.7849999964
	1.0000000000	0.8679136485	0.0010905191
3	0.0000000000	0.7849999964	0.7849999964
	1.0000000000	0.8875938505	0.0400029011
4	0.0000000000	0.7849999964	0.7849999964
	1.0000000000	0.8886059821	0.0224689320
5	0.0000000000	0.7849999964	0.7849999964
	1.0000000000	0.8837629259	0.0077346563
6	0.0000000000	0.7849999964	0.7849999964
	1.0000000000	0.8835852146	0.0092189689
7	0.0000000000	0.7849999964	0.7849999964
	1.0000000000	0.8829277456	0.0078043751
8	0.0000000000	0.7849999964	0.7849999964
	1.0000000000	0.8827768266	0.0072949543
9	0.0000000000	0.7849999964	0.7849999964
	1.0000000000	0.8813270479	0.0067779524
10	0.0000000000	0.7849999964	0.7849999964
	1.0000000000	0.8813672960	0.0066111484
11	0.0000000000	0.7849999964	0.7849999964
	1.0000000000	0.8814257830	0.0067376704
12	0.0000000000	0.7849999964	0.7849999964
	1.0000000000	0.8814252317	0.0067664690
13	0.0000000000	0.7849999964	0.7849999964
	1.0000000000	0.8814192563	0.0067955683
14	0.0000000000	0.7849999964	0.7849999964
	1.0000000000	0.8814185113	0.0067782488

TABLE 17

EULER METHOD			
K=1	t	x_1	x_2
	0.0000000000	0.7849999964	0.7849999964
	1.0000000000	0.9426864088	-0.4416966140
2	0.0000000000	0.7849999964	0.7849999964
	1.0000000000	0.9193312973	-0.1712644026
3	0.0000000000	0.7849999964	0.7849999964
	1.0000000000	0.9185541421	-0.0245606415
4	0.0000000000	0.7849999964	0.7849999964
	1.0000000000	0.9033740759	-0.0134939626
5	0.0000000000	0.7849999964	0.7849999964
	1.0000000000	0.8915344328	-0.0119556952
6	0.0000000000	0.7849999964	0.7849999964
	1.0000000000	0.8868568391	-0.0003889427
7	0.0000000000	0.7849999964	0.7849999964
	1.0000000000	0.8847047687	0.0031411243
8	0.0000000000	0.7849999964	0.7849999964
	1.0000000000	0.8837036937	0.0050881854
9	0.0000000000	0.7849999964	0.7849999964
	1.0000000000	0.8817466199	0.0055904603
10	0.0000000000	0.7849999964	0.7849999964
	1.0000000000	0.8815876842	0.0060214628
11	0.0000000000	0.7849999964	0.7849999964
	1.0000000000	0.8815371841	0.0064465657
12	0.0000000000	0.7849999964	0.7849999964
	1.0000000000	0.8814813346	0.0066198139
13	0.0000000000	0.7849999964	0.7849999964
	1.0000000000	0.8814466149	0.0067221449
14	0.0000000000	0.7849999964	0.7849999964
	1.0000000000	0.8814319670	0.0067414588

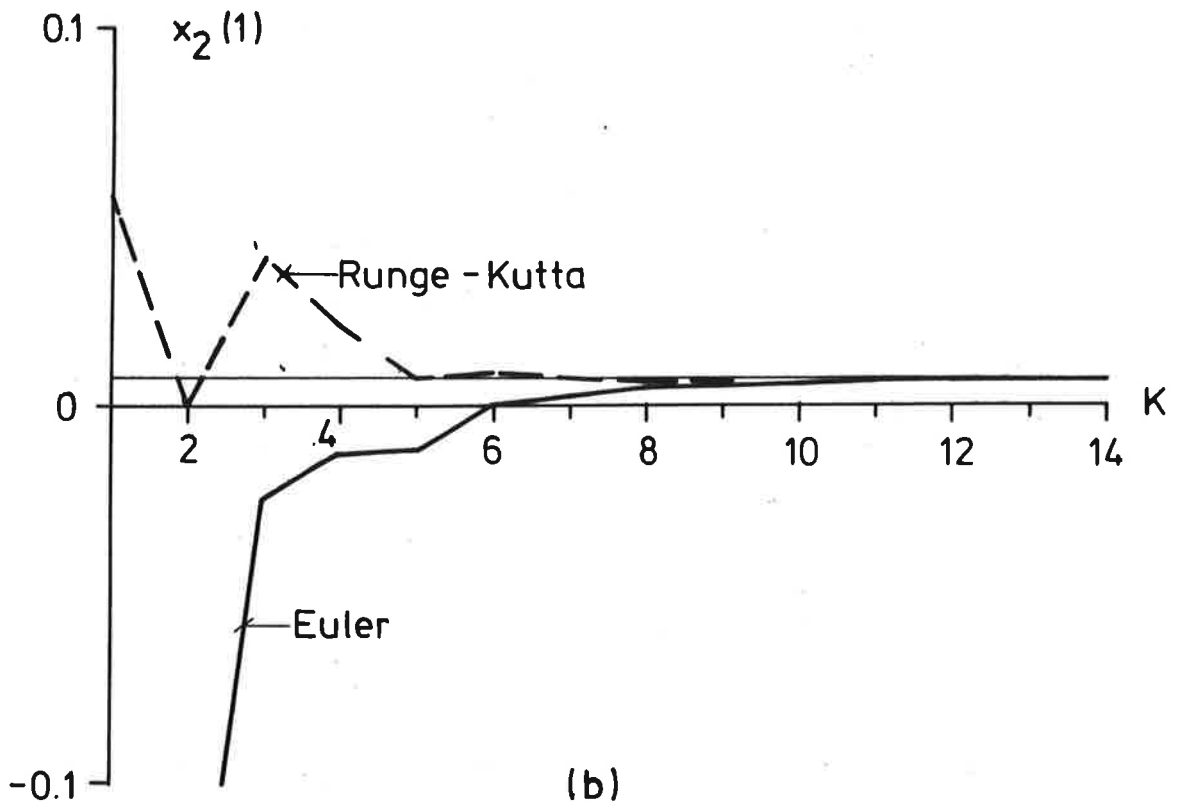
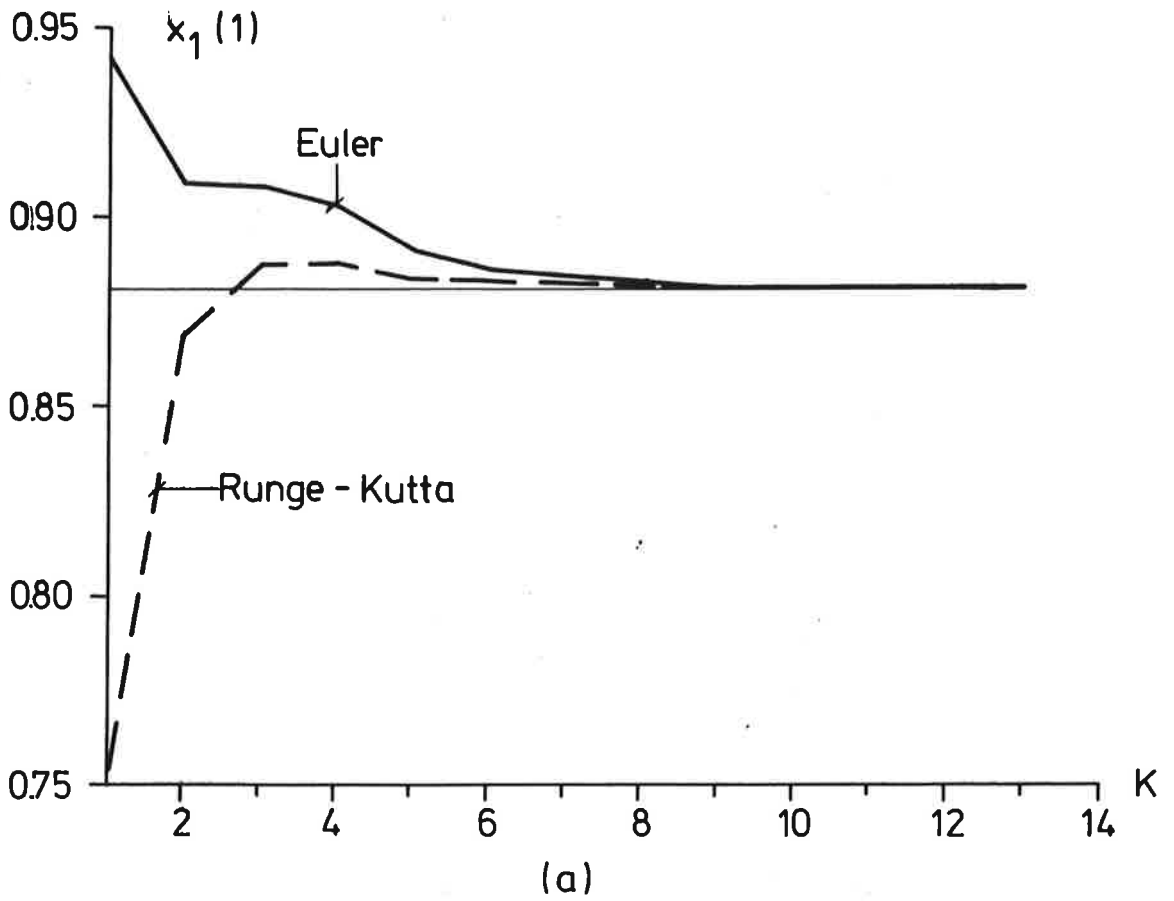


Fig.7. Phase-lock loop (41)

7. ACKNOWLEDGEMENTS

The author would like to express his gratitude to forskn. ing. Thomas Schönthal for the consultations on programming, to Miss Britt-Marie Carlsson for beautiful drawing of the pictures, and to Mrs Eva Dagnegård for help with the English and for careful typewriting.

8. REFERENCES

1. Doob J H: Stochastic Processes. Wiley, New York, 1953.
2. Cumming I G: Computing Aspects of Problems in Non-linear Prediction and Filtering. PhD Thesis, University of London, 1967.
3. Brahic A, Bourret R: Stochastically Perturbed Differential Equations: The Need and Possibility of Numerical Solution. Rev CETHEDC 10, No 34 (1973).
4. Mc Shane E J: Stochastic Calculus and Stochastic Models. Academic Press, New York, 1974.
5. Wright D J: The Digital Simulation of Stochastic Differential Equations. IEEE Trans Automatic Control AC-19, No 1 (1974) 75-76.
6. Rao N J, Borwankar J D, Ramkrishna D: Numerical Solution of Ito Integral Equations. SIAM J Control 12, No 1 (1974) 124-139.
7. Nikitin N N, Pervachev S V, Razevig V D: On Computer Solution of Stochastic Differential Equations for Follow-up Systems. Automation and Remote Control, No 4 (1975) 133-137.
8. Taylor F J: On Stochastic Phase-lock Loop Solutions. IEEE Trans Communications, January (1976) 138-140.
9. Nikitin N N, Razevig V D: Methods of the Digital Simulation of Stochastic Differential Equations and Estimation of Their Accuracy. To appear in the USSR Computer Mathematics and Mathematician Physics, No 2 (1978).
10. Jazwinski A H: Stochastic Processes and Filtering Theory. Academic Press, New York-London, 1970.

11. Gear C W: Numerical Initial Value Problems in Ordinary Differential Equations. Prentice-Hall, Inc, Englewood Cliffs, New Jersey, 1971.
12. Jansson B: Random Number Generators. Victor Pettersons Bokindustri AB, Stockholm, 1966.
13. Programming and Documentation Rules for Subroutine Libraries - Designed for the SCL. Eds Elmqvist, Tyssø, Wieslander, 1976.
14. Harris C J: Modelling, Simulation and Control of Stochastic Systems with Applications in Wastewater Treatment. Int J Systems Sci 8, No 4 (1977) 393-411.
15. Harris C J: Control Systems Centre Technical Report No 313, UMIST, England, 1976.

9. APPENDIX

Iterative solution of the n-dimensional Itô stochastic differential equation (19) provides the next difference scheme, similar to the one-dimensional scheme (15):

$$\begin{aligned}
 x_i(t) = & x_i(t_0) + a_i \Delta t + \sum_{j=1}^m b_{ij} \Delta w_j(t_0) + \\
 & + \sum_{j=1}^m \sum_{k=1}^n \sum_{\ell=1}^m \frac{\partial b_{ij}}{\partial x_k} b_{k\ell} \eta_{\ell j}(t_0) + \frac{\Delta t}{2} \sum_{j=1}^m \left(\frac{\partial b_{ij}}{\partial t} + \right. \\
 & + \left. \sum_{k=1}^n \frac{\partial a_i}{\partial x_k} b_{kj} + \sum_{k=1}^n \frac{\partial b_{ij}}{\partial x_k} a_k \right) \Delta w_j(t_0) + \\
 & + \left(\frac{\partial a_i}{\partial t} + \sum_{k=1}^n a_k \frac{\partial a_i}{\partial x_k} \right) \frac{(\Delta t)^2}{2}
 \end{aligned} \tag{41}$$

where $\Delta w_j(t_0) = w_j(t) - w_j(t_0)$,

$$\eta_{\ell j}(t_0) = \begin{cases} \frac{1}{2} [(\Delta w_j(t_0))^2 - \Delta t], & \ell = j, \\ \frac{1}{2} \Delta w_\ell(t_0) \Delta w_j(t_0), & \ell \neq j. \end{cases} \tag{42}$$

Algorithm (41) is not used in our routine. We have published it, as expression (41) is more general than expressions (15) and (23) in [14]. Our scheme (41) provides not only sample moments convergence as (15) and (23) in [14], but also the sample path convergence. Besides expression (23) in [14] contains some misprints and expression (15) in [14] doesn't consider the correlation between dependent random variables Z_{1n} and Z_{2n} . Such a correlation is taken into account in scheme (3.2) in [6] but this algorithm is implementable only for the one-dimension case (and provides only sample moments convergence).

In order to obtain the Stratonovich solution of equation (19) by means of (41) it is necessary to replace expression (42) by

$$\eta_{\ell j}(t_0) = \frac{1}{2} \Delta w_{\ell}(t_0) \cdot \Delta w_j(t_0). \quad (42)$$