Computer Algebra and Stochastic Analysis

Some Possibilities

E. Valkeila
Computing Centre, University of Helsinki
Teollisuuskatu 23, SF-00510 Helsinki, Finland
valkeila@cc.helsinki.fi

The possible applications of computer algebra to stochastic analysis are discussed. We show some variants of the change of variables formulas for different kinds of stochastic processes. The computer algebra program we use is Macsyma.

1 Introduction

1.1. Suppose that f is a real function with a derivative f_x and $f(0) = 0$. Then we have from elementary analysis that

$$f(t) = \int_0^t f_x(s)ds. \tag{1.1}$$

Now we want to replace ds in (1.1) by dW_s, where W is the Wiener process. Recall that the paths of W have unbounded variation but finite quadratic variation on compact intervals. If f_{xx} exists, the Ito formula (the change of variables formula for the Wiener process) says that then we have the following expression for the process $f(W_t)$:

$$f(W_t) = \int_0^t f_x(W_s)dW_s + \frac{1}{2}\int_0^t f_{xx}(W_s)ds. \tag{1.2}$$

The first integral in (1.2) is a stochastic integral, which can be defined using the fact that the paths of W have bounded quadratic variation, and the second integral is the usual Lebesgue-Stieltjes integral. Below we show how computer algebra can be used to generate the right-hand side of (1.2). We shall also study the analogue of (1.1) for counting processes, i.e. for processes with piecewise constant increasing paths with unit jumps.

We assume that the reader is familiar with the Ito formula and the change of variables formula for counting processes. Below we concentrate on the computational or algebraic aspects of stochastic analysis. We have no discussion at all of technical matters such as the construction of stochastic integrals, or questions like "is some expression a martingale or only a local martingale?" etc.

We recommend [6] or [8] to be consulted on such questions.

1.2. Kendall [3,4] has systematically worked on computer algebra and stochastic analysis. In Section 3 we describe his work in greater detail. We list some applications of computer algebra to statistics and probability. Ruskeepää [10] used Macsyma to compute the exact prediction formula for certain AR(1)-processes. Rehak et al. [9] list the following examples: cumulants of a distribution, deterministic and random oscillators excited by a random force, random vibration of a simply supported beam and equations with white noise random coefficients. They use Macsyma. Keady [2] explains how computer algebra can be used in maximum likelihood estimation. Kendall [5] states that his computerized stochastic analysis can be applied to simulation of continuous semimartingales.

1.3. This paper is organized as follows. In the next section we give a brief description of the capabilities of computer algebra programs. In the third section we show how the two kinds of formulas can be programmed using Macsyma. In the last section we try to illustrate how the two Ito formulas can applied to some problems.

2 On computer algebra

2.1. Computers can be used to do symbolic computations. Over thirty years ago the computer was first used to perform algebraic calculations, e.g. to differentiate simple expressions ([1]). Today there exist programs like Macsyma, Maple, Reduce and Mathematica, which are very useful for a working mathematician and scientist. In the next section we list some common features of these programs.

2.2. The existing computer algebra systems usually are provided with the following tools:

- Exact computations on integers, rational, real and complex numbers with *unlimited accuracy.*
- Operations on polynomials in one or more variables and on rational fractions. In short, the obvious rational operations, calculating the g.c.d., factorizing over integers.
- Calculations with matrices with numerical and/or symbolic elements.
- Simple analysis: differentiation, expansion in series, Padé approximations, etc.
- Manipulation of formulas: various substitutions selection of coefficients and of parts of the formulas, numerical evaluation, pattern recognition, controlled simplifications.

These allow the user to do complicated calculations within some minutes (see [11], Chapter 3). In contrast to calculations done manually, computer algebra calculations are more reliable.

Based on the above tools the systems also offer:

- Solution of equations, formal integration, calculation of limits, tensor calculus.

The systems usually have the following features and possibilities:

- Use the system as a programming language, graphical facilities, generation of program code as parts of numerical programs.

2.3. Below we show a computer algebra session. The (Ci) lines are input command lines and the (Di) lines are the output. We use Macsyma at the University of Helsinki Computing Centre implemented on a VAX 8800 computer.

```
(C7) S:0 $
(C8) FOR I THRU 37 DO S:S + 1 / (I **2);
(D8)                                  DONE
(C9) S;                  46196589536413702085491232689909
(D9)                     --------------------------------
                         285469165548754893851168794240000
(C10) DIFF(SIN(X)*COS(X),X);
                                    2          2
(D10)                            COS (X) - SIN (X)
(C11) TRIGSIMP(COS(X)**2-SIN(X)**2);
APPL_DISK:[Macsyma_DIR.Macsyma_412.SHARE]trgsmp.fas;1being loaded.
                                        2
(D11)                                2 COS (X) - 1
```

On lines (C7)-(C9) we compute the sum $\Sigma_{i=1}^{37} i^{-2}$. Note the recursive definition of the sum and exact computations. On lines (C10)-(C11) we compute the first derivative of the function $f(x) = \sin(x)\cos(x)$ and simplify the result. We refer to [1] for a more detailed introduction to computer algebra. For a tutorial in Macsyma we refer to [12].

It has been estimated that the most frequent application of computer algebra is in automatic code generating [11].

3 Stochastic analysis with the computer

3.1. The main tool of stochastic analysis is the change of variables formula. We recall formula (1.2):

$$f(W_t) = \int_0^t f_x(W_s)dW_s + \frac{1}{2}\int_0^t f_{xx}(W_s)ds.$$

If expectations can be taken on both sides of this equality, we have the following

$$Ef(W_t) = \tfrac{1}{2}E\int_0^t f_{xx}(W_s)ds. \tag{3.1}$$

It is possible to prove that (3.1) is true for stopping times T instead of the fixed times t. This explains why we are usually more interested in the term $\int_0^t f_{xx}(W_s)ds$ than in the first term on the right-hand side of (1.2).

3.2. Next we show how one can program the formula (1.2) into Macsyma. This is done by the following lines:

```
ito(F):=block( [w1,w2],
w1:ratsimp(diff(F,'x)),
w2:ratsimp(diff(F,'x,2)/2),
ito_list : [subst(``W(t)'',x,F),
ev(subst(0,x,F)),
subst(``W(t)'' ,x,w1),
subst(``W(t)'',x,w2)] ) $
```

The above program is relatively simple. We use the existing Macsyma-commands for differentiation and then simplify the expression using the simplification command `ratsimp`. Then we return the result into the list `ito_list` with four elements: $f(W_t), f(0), f_x(W_t)$ and $f_{xx}(W_t)$. The user calls this program by typing the command `ito(f(x));`, where f is a function of x. The following lines are almost an exact output of the corresponding Macsyma-session. Before this, we have loaded the contents of the above small program from disk to our Macsyma-session.

```
(C8) ITO(X**3);
                            3            2
(D8)                  [W(t) , 0, 3 W(t) , 3 W(t)]
(C10) ITO(LOG(1+X**2));
                                                               2
                         2            2 W(t) ,            W(t) -1
(D10)         [LOG(W(t)  + 1), 0, ----------- - ------------------ ]
                                          2                4          2
                                     W(t) + 1      W(t)  + 2 W(t)  + 1
```

From the above we have the following formulas:

$$W_t^3 = 3\int_0^t W_s^2 dW_s + 3\int_0^t W_s ds$$

and

$$\log(W_t^2+1) = 2\int_0^t \frac{W_s}{W_s^2+1}dW_s - \int_0^t \frac{W_s^2-1}{W_s^4+2W_s^2+1}ds.$$

It took 50 milliseconds of cpu-time to compute the first formula and 170 milliseconds to compute the second formula (see [7], 17-19). Note also how the computer input and output above looks. We expect that during the next few years this will be greatly improved, perhaps by using other programs as filters for the input and output.

The above stochastic calculus can be extended to a continuous semimartingale of the form $X_t = \int a(X_s)ds + \int_0^t b(X_s)dW_s$, with $X_0 = 0$. The corresponding

program is listed in the Appendix. We illustrate the use of this program by proving the Girsanov formula, which in this case means that if X satisfies

$$X_t = -\frac{1}{2}\int_0^t b_s^2(X_s)ds + \frac{1}{2}\int_0^t b_s(X_s)dW_s$$

then the process e^{X_t} is a martingale.

```
(C20) DIFF_ITO([%E**X,-B**2/2,B]);
                                          X(t)          X(t)
(D20)                                [%E     , 1, B %E     , 0]
```

Note the use of the function on line (C20): the argument is a list with the function f, drift coefficient and diffusion coefficient. The result tells us that the drift term is equal to 0, so the process is a martingale if the integrability conditions can be checked. To remind, the output in (D20) corresponds to the following equation

$$e^{X_t} = 1 + \int_0^t b(X_s)e^{X_s}dW_s,$$

so the result is a list similar to that above.

As noted before, Kendall [3,4,5] has already used computer algebra for stochastic analysis. He has used the computer algebra program Reduce. Our approach is quite direct—we program some functions and then the user can call these functions to perform the computations. Kendall defines stochastic differentials using a so-called Ito multiplication table ($dtdW_t = 0, (dt)^2 = 0$ and $(dW_t)^2 = dt$) and then builds his computerized stochastic calculus on this information.

We shall give a simple example for multidimensional diffusion processes in the last section.

3.3. Next we turn to the other kind of processes. We say that the process is cadlag, if its paths are continuous from the right and has left limits. The process, denote it by N, is a counting function: $N_0 = 0, \Delta N \in \{0, 1\}$ and N is a constant between the jumps. Then, if f is a C_1 - function, we have

$$f(N_t) = f(0) + \int_0^t f_x(N_{s-})dN_s + \sum_{s\le t}(\Delta f(N_s) - f_x(N_{s-})\Delta N_s), \tag{3.2}$$

where f_x is as above and $\Delta X_t = X_t - X_{t-}$ is the jump at time t for a cadlag function. The following is an alternative formula for $f(N_t)$:

$$f(N_t) = f(0) + \sum_{s\le t}\Delta f(N_s). \tag{3.3}$$

The problem is to write such a representation for $f(N_t)$ that one can compensate $f(N_t)$ to a martingale or write a multiplicative decomposition for $f(N_t)$. The following lemma shows a typical situation in this context. If B is a function with bounded variation, denote by $\mathcal{E}(B)$ the Dolean exponent $\mathcal{E}_t(B) =$

$\exp(B_t)\Pi_{s\leq t}((1+\Delta B_s)\exp(-\Delta B_s))$. Recall that the compensator A of N is the unique predictable process s.t. $N-A$ is a martingale.

LEMMA 3.1 *Suppose that N is a counting process with compensator A. Then, if we have for $f(N_t)$ the following representation*

$$f(N_t) = 1 + c\int_0^t f(N_{s-})dN_s,$$

where c is a constant, then $f(N_t)$ has the following representation

$$f(N_t) = f(N_0)\mathcal{E}(cA)\mathcal{E}(c\tilde{m}), \tag{3.4}$$

where $\tilde{m} = (1+\Delta A)^{-1}\circ(N-A)$.

PROOF. See [6] Theorem II.5.1.1. for a more general result, from which (3.4) follows. $\diamondsuit$

To illustrate the use of Lemma 3.1, assume that the compensator A is a deterministic function. Then we have from (3.4) (assuming again that integrability conditions are checked):

$$Ef(N_t) = f(0)\mathcal{E}_t(cA).$$

As an example we compute the Laplace transform of a counting process N with a deterministic compensator A.

First we compute the expression for $\exp\{\lambda N_t\}$:

```
(C36) POISS1 (%E**(L*X));
                                  L N(t)          L          L N(t-)
(D36)                           [%E       , 1, (%E   - 1) %E       ]
```

So the equation for $f(N_t) = \exp\{\lambda N_t\}$ is

$$f(N_t) = 1 + \int_0^t f(N_{s-})\exp\{\lambda - 1\}dN_s$$

and the Laplace transform is

$$E\exp\{\lambda N_t\} = \mathcal{E}_t(\exp\{\lambda - 1\}A).$$

The Macsyma - program for this is listed below:

```
poiss1(F) :=
blocl([w1,w2,w3,w4,w5],
w1 :  subst (i+1,x,F),
w2 :  subst(i, x, F),
w3 :  (w1-w2),
w4 :  radcan(w3/w2),
w5 :  subst (''N(t-)'',i,w2) *w4,
comp_list :   [subst(''N(t)'', i+1,w1),
ev(subst(0,x,F)),
w5 ]
)$
```

After differentiating the function we try to simplify the sum containing the jumps to a form where we can apply Lemma 3.1. This is not always possible. So it is sometimes useful to try to simplify the second sum in (3.2). Our second program for counting processes exploits this idea. The program is listed in the Appendix. We give an example of its use in the next section.

4 Examples

4.1. First we consider the multi-dimensional diffusion process case. It is clear that the more computations we have, the more useful computer algebra is in stochastic analysis. We give only a simple example. Suppose that we want to compute

$$A_t W_t = \int_0^t A_s dW_s + \int_0^t W_s dA_s, \tag{4.1}$$

where a is a continuous function with bounded variation and W is the Wiener process. We assume that $A_t = \int_0^t a_s ds$. Then, if we consider the process $X_t = (W_t, A_t)'$, it is a two-dimensional diffusion process with coefficients $\alpha_t = (0, a_t)'$ and $\beta_t = (1,0)'$ so that $X_t = \int_0^t \alpha_s ds + \int_0^t \beta_s dW_s$. Then we obtain (4.1) by applying the function $f(x_1, x_2) = x_1 x_2$ to X. The computer output is:

```
(C11) AA: [0,A];
(D11)                                          [0,A]
(C12) BB: [1,0];
(D12)                                          [1,0]
(C13) F:X1*X2;
(D13)                                          X1 X2
(C14) MULTI_ITO ([F,AA,BB]);
(D14)                          [X1 X2 , 0, DX1 X2, AX1]
```

On lines (C11)-(C13) we give the information for the coefficients α and β. The result is a list (multi_ito_list) containing four elements: the function, initial value, martingale part, and trend part. Here we have $X1 = W_t, X2 = A_t$ and $AX1 = a_t W_t$. So again we are facing the limitations of the input/output. Note that we write the 'differential' in the martingale part. This should be read as $DX1 = dW_t$, since we have only one Wiener process here. For more serious use of computerized stochastic analysis we refer to [3] and [4].

4.2. We conclude by giving an example of formula (3.2). Assume that the counting process N has independent increments, so the compensator A is an increasing deterministic function, but not necessarily continuous. We compute the moments EN_t^k for $k = 2$ and $k = 3$.

Recall the integration by parts formula

$$U_t V_t = U_0 V_0 + \int_0^t U_{s-} dV_s + \int_0^t V_{s-} dU_s + [U, V]_t$$

for functions U and V with bounded variation, where $[U, V]_t = \sum_{s \le t} \Delta U_s \Delta_s$. Returning to our problem, here is the computer output:

```
(C24) POISS (X**2);
                                  2
(D24)                    [N(t)  , 0, 2 N(t-), N(t)]
(C25) POISS (X**3);
                          3              2  N(t) (3 N(t) - 1)
(D25)              [N(t)  , 0, 3 N(t-)  , ---------------------]
                                                  2
```

Put $m_k(t) = EN_t^k$. We have $m_1(t) = A_t$ and from (D24) (since A is deterministic, and by Fubini)

$$m_2(t) = 2\int_0^t m_1(s-)dA_s + m_1(t) = 2\int_0^t A_{s-}dA_s + A_t$$

i.e.

$$m_2(t) = A_t^2 + A_t - [A, A]_t. \tag{4.2}$$

From (D25) we have that

$$m_3(t) = 3\int_0^t m_2(s-)dA_s + \frac{3m_2(t) - m_1(t)}{2}, \tag{4.3}$$

and this gives

$$m_3(t) = A_t^3 + 3A_t^2 + A_t + 2[A, [A, A]]_t - 3A_t[A, A]_t. \tag{4.4}$$

If A is continuous, we have the familiar formulas $m_2(t) = A_t^2 + A_t$ and $m_3(t) = A_t^3 + 3A_t^2 + A_t$.

To obtain (4.4) from (4.3) we use (4.2) and integration by parts (done manually, not by computer). We feel that teaching the computer to perform the above computations involves considerable difficulty and so there is still much work to do.

References

1. J.H. Davenport, Y. Siret, E. Tournier (1988). *Computer Algebra—Systems and Algorithms for Algebraic Computation*, Academic Press, New York.
2. G. Keady. *SENAC and Other Symbolic Front Ends for Subsequent Numerical Computation: Nonlinear Systems and Optimization Case Studies*, University of Waikato, Research Report Series, II no. 11.
3. W. Kendall (1988). Symbolic computation and the diffusion of shapes of triads. *Adv. Apll. Prob. 20*, 775-797.
4. W. Kendall (1990). The diffusion of Euclidean Shape, In: *Disorder in Physical Systems*, ed. by Grimmet, G. and Welsh, D., Oxford University Press, Oxford, 203-217.
5. W. Kendall (1990). Computer Algebra and Stochastic Calculus. *Notices Am. Math. Soc., 37*, 1254-1256.

6. R.S. LIPSTER, A.N. SHIRYAEV (1989). *Theory of Martingales*, Kluwer, Amsterdam.
7. *Macsyma Reference Manual*, Symbolics 1988.
8. P. PROTTER (1990). *Stochastic Integration and Differential Equations—A New Approach*, Springer.
9. M.L. REHAK, F.L. DIMAGGIO, H. BENAROYA, I. ELISHAKOFF (1987). Random vibrations with Macsyma. *Computer Methods in Applied Mechanics and Engineering 61*, 61-70.
10. H. RUSKEEPÄÄ (1988). Exact predictors for a generalized AR(1) process with an AR(1) parameter. *Commun. Statist.-Theory Meth., 17*, 875-885.
11. Future directions for research in symbolic computation, *SIAM Reports on issues in the Mathematical Sciences* 1990.
12. *Macsyma User's Guide*, Symbolics, 1988.

APPENDIX

The program for diffusion processes

```
diff_ito(fns) :=block( [w1,w2],
w1 :ratsimp(fns[3]*diff(fns[1],'x)),
w2:ratsimp(fns[3]**2 * diff(fns[1],'x,2)/2+fns[2]*diff(fns[1],'x)),
ito_list : [ subst(''X(t)'',x,fns[1]),
ev(subst(0,x,fns[1])),
subst(''X(t)'',x,w1),
subst(''X(t)'',x,w2) ]) $
```

The program for counting processes

Below is the program for the counting processes to compute formula (3.2). We try to write the formula $\sum_{s\le t}(\triangle f(N_s) - f_x(N_{s-})\triangle N_s)$ in a closed form. We use the simplification function closedform for finite sums, which is contained in the nusum1 package of Macsyma.

```
poiss(F) :=
    block([a,w1,w2,w3,w4],
    a : subst(i-1,x,diff(f,'x)),
    w1 : subst(i,x,F),
    w2 : subst(i-1,x,F),
    w3 : (w1-w2) -a1,
    w4 : closedform(sum(w3,i,1,n)),
    poiss_list:
       [ subst(''N(t)'',x,f),
          ev(subst(0,x,F)),
          subst(''N(t-)'',i-1,a),
          subst(''N(t)'',n, w4) ]
)$
```

The program for multi-dimensional diffusion processes

```
multi_ito(fns):=block ( [w0,w1,w2,w3,w4,i,n,xx,dx],
n : length(fns[2]),
xx : makelist(concat(x,i),i,1,n),
dx : makelist(concat(dx,i),i,1,n),
for i : 1 thru n do
    w1[i] : ratsimp( diff(fns[1], xx[i])),
for i : 1 thru n do
    w2[i] : ratsimp (diff(fns[1],xx[i], 2)/2),
w3 : ratsimp(sum( w1[i]*fns[3][i]*dx[i],i,1,n) ),
w4 :ratsimp(sum((w1[i]*fns[2][i] + w2[i]*fns[3][i]**2), i,1,n)),
w0 : fns[1],
for i : 1 thru n do
w0 : ev( subst(0,xx[i],w0) ),
multi_ito_list : [ fns[1],
w0,
w3,
w4 ] )$
```