

Maple Schemes for Jump-Diffusion Stochastic Differential Equations

Sasha Cyganowski

School of Computing and Mathematics, Deakin University
Waurm Ponds, 3217, Australia
e-mail: sash@deakin.edu.au

Peter Kloeden

Fachbereich Mathematik, Johann Wolfgang Goethe Universitat
D-60054 Frankfurt am Main, Germany
e-mail: kloeden@math.uni-frankfurt.de

Introduction

Deterministic differential equations provide adequate models of dynamical systems when intrinsic noise within the system is low. To obtain a more detailed model of a noisy dynamical system a stochastic model is often used. A general, one-dimensional Stochastic Differential Equation (SDE) has the form

$$dX_t = a(t, X_t)dt + b(t, X_t)dW_t, \quad (1)$$

where $a(t, X_t)$ is the drift coefficient and $b(t, X_t)$ is the diffusion coefficient. The random term, W_t , is a Wiener process. The “derivative” of the Wiener process is often referred to as white noise. The subscript t denotes time-dependence. The Wiener process is the simplest random or stochastic process which provides an adequate model for Brownian motion. Further details of the Wiener process and methods for stochastic differential equations can be found in Kloeden and Platen [1].

Stochastic Differential Equations have been used extensively in areas such as Chemistry, Physics, Engineering, Mathematical Biology and Finance to provide models of dynamical systems affected by noise. When it is the case that a stochastic system is also affected by randomly occurring impulses it is often desirable to use a jump-diffusion stochastic model. A general, n -dimensional jump-diffusion SDE has the form

$$dX_t = A(t, X_t)dt + B(t, X_t)dW_t + C(t, X_{t-})dN_t, \quad (2)$$

where X_t is an n -dimensional vector of the solution, $A(X_t)$ is the n -dimensional drift vector, $B(X_t)$ is the diffusion matrix of order $n \times m$, and $C(X_{t-})$ is an n -dimensional vector. In equation (2), W_t is m -dimensional, and N_t is the inhomogeneous Poisson counting process.

It is interesting to note that the Wiener process is not of bounded variation, thus if we write equation (2) as an integral equation of the form

$$\begin{aligned}
X_t(w) = & X_{t_0}(w) + \int_{t_0}^t a[s, X_s(w)] ds + \int_{t_0}^t b[s, X_s(w)] dW_s(w) \\
& + \int_{t_0}^t c[s, X_{s-}(w)] dN_s(w), \tag{3}
\end{aligned}$$

the second integral is not defined by the rules of classical calculus. This problem was overcome in the early 1950s when Ito formulated his definition of the Ito integral. The third integral in equation (3) is defined as a stochastic integral with respect to a Poisson random measure (Gikhman and Skorokhod [2], Grigelionis [3]). It follows from Ito [4] and Gikhman and Skorokhod [2] that the solution process X_t to equation (3) satisfies existence and pathwise uniqueness under growth restriction, uniform Lipschitz, and smoothness conditions on the functions a , b and c .

Since dynamical systems modeled by jump-diffusion stochastic differential equations such as equation (2) and ordinary stochastic differential equations such as equation (1) rarely possess known explicit solutions, numerical analysis of such equations has become a rapidly growing area. Time-discrete numerical schemes for ordinary stochastic differential equations were constructed from stochastic Taylor expansions based on the Ito formula, or stochastic chain rule, by Milshtein [5] and Kloeden and Platen [1].

Constructing such numerical schemes is made easier through the use of a symbolic manipulator. This is especially true for higher order schemes, which often become algebraically large. The computer algebra system Maple has proved a useful tool for the implementation of stochastic numerical schemes (see Cyganowski [6], Cyganowski [7], Cyganowski and Kloeden [8], Kloeden and Scott [9] and Xu [10]). Stochastic differential equations have also been studied through the use of other computer algebra systems by Kendall [11] and Valkeila [12].

A search of the literature reveals that no computer algebra representation exists for the numerical methods available for jump-diffusion SDEs of the form of equation (2). In the paper by Maghsoodi [13], numerical schemes are

derived for jump-diffusion SDEs which are proved to converge with order 2 in the mean square sense. A scheme which converges in practice with order 3 in the mean square sense (for a certain class of jump-diffusion SDE only) is also presented. In this paper, Maple routines are presented which implement the schemes presented by Maghsoodi [13].

Construction of Numerical Schemes with Maple

Consider the scalar equation

$$X_t = X_{t_0} + \int_{t_0}^t a(s, X_s) ds + \int_{t_0}^t b(s, X_s) dW_s + \int_{t_0}^t c(s, X_s) dN_s, \quad (4)$$

where $X_{t_0} = X_0$ with probability one (w.p.1) and $t \in [t_0, T]$. Let Y_n approximate X_t at t_n and let ΔW_{n+1} and ΔN_{n+1} represent the increments of the Wiener process and the Poisson process, respectively, over the interval (t_n, t_{n+1}) .

With this notation, the simplest numerical scheme, the stochastic Cauchy-Euler scheme, is given by

$$Y_{n+1} = Y_n + ah + b\Delta W_{n+1} + c\Delta N_{n+1} \quad (5)$$

for $n = 0, 1, \dots, M - 1$ where a, b and c are evaluated at (t_n, Y_n) . It is established in Maghsoodi [13] that this scheme is $O(h)$ in the mean square sense.

Figure 1 belows displays a Maple procedure which returns stochastic Cauchy-Euler schemes, as defined by equation (5).

```
sce:=proc(a::algebraic,b::algebraic,c::algebraic)
local soln,h;
soln:=Y[n+1]=Y[n]+a*h+b*W[n+1]+c*N[n+1];
soln:=subs(x=Y[n],soln)
end;
```

Figure 1 Procedure ‘sce’.

The syntax for procedure ‘sce’ is shown below in figure 2.

```
sce(a,b,c);
```

Figure 2 Syntax for ‘sce’.

In figure 2, the input functions a, b and c are required to be functions of a variable x.

In the multi-dimensional case, the functions X_t , a and c in equation (2) become $n \times 1$ matrices and b becomes an $n \times m$ matrix. Thus, equation (5) may be extended to give the multi-dimensional, stochastic Cauchy-Euler scheme, as shown below.

$$Y_{n+1}^k = Y_n^k + a^k h + \sum_{j=1}^m b^{k,j} \Delta W^j + c^k \Delta N. \quad (6)$$

Here, $\Delta W^j = W_{t_{n+1}}^j - W_{t_n}^j$ is the $N(0; h)$ distributed increment of the j th component of the m -dimensional standard Wiener process W on (t_n, t_{n+1}) , and ΔW^{j_1} and ΔW^{j_2} are independent for $j_1 \neq j_2$.

Figure 3 below displays a Maple procedure which returns stochastic Cauchy-Euler schemes for multi-dimensional equations of the form (2).

```
sceMD:=proc(a::array,b::array,c::array)
local i,u,soln,h;
for i to rowdim(a) do
soln[i]:=Y.i[n+1]=Y.i[n]+a[i,1]*h+sum('b[i,j]*W.j[n+1]',j'=1..coldim(b))
+c[i,1]*N[n+1];
for u to rowdim(a) do soln[i]:=subs(x[u]=Y.u[n],soln[i]) od
od;
RETURN(eval(soln))
end;
```

Figure 3 Procedure ‘sceMD’.

Note that the input variables a, b, and c in procedure ‘sceMD’ must be matrices of appropriate order. Thus, the Maple package ‘linalg’ must be initially read into the worksheet. Also, any variables present in the elements of the matrices must be given in the form $x[1]$, $x[2]$, ... , $x[n]$, where n is the

dimension of the system.

Maghsoodi [13] has derived schemes that are improvements in order, in the mean square sense, on the above Cauchy-Euler scheme. In generalizing the approach that Milshstein [5] used to derive efficient second order schemes for stochastic differential equations (of the form (1)), Maghsoodi [13] has derived the following schemes, which converge to their respective orders in the mean square sense.

Second Order Schemes

The numerical scheme given below is of second order in the mean square sense (Maghsoodi [13]), and is suitable for scalar jump-diffusion equations.

$$\begin{aligned}
Y_{n+1} = & Y_n + (a - \frac{1}{2}bb_x)h + b\Delta W_{n+1} + \frac{1}{2}bb_x\Delta W_{n+1}^2 \\
& + \frac{1}{2}(3c - c_c)\Delta N_{n+1} + (b_c - b)\Delta W_{n+1}\Delta N_{n+1} \\
& + \frac{1}{2}(c_c - c)\Delta N_{n+1}^2 + (bc_x - b_c + b)\Delta Z_{n+1}, \quad (7)
\end{aligned}$$

where a subscript x denotes partial differentiation with respect to x , $b_c = b(t, x + c)$,

$$\Delta Z_{n+1} = \int_{t_0}^{t_{n+1}} (W_s - W_k) dN_s \quad (8)$$

and all functions are evaluated at the point (t_n, Y_n) .

Figure 4 below displays a Maple procedure which returns second order schemes of the form (7) for scalar dump-diffusion equations.

```

jump2:=proc(a::algebraic,b::algebraic,c::algebraic)
local soln;
soln:=Y[n+1]=Y[n]+(a-(1/2)*b*diff(b,x))*h+b*W[n+1]
+(1/2)*b*diff(b,x)*(W[n+1])^2+(1/2)*(3*c-subs(x=Y[n]+c,c))*N[n+1]
+(subs(x=Y[n]+c,b)-b)*W[n+1]*N[n+1]+(1/2)*(subs(x=Y[n]+c,c)-c)*(N[n+1])^2
+(b*diff(c,x)-subs(x=Y[n]+c,b)+b)*Z[n+1];
subs(x=Y[n],soln)
end:

```

Figure 4 Procedure ‘jump2’.

The syntax for the procedure ‘jump2’ is identical to that for the procedure ‘sce’ (see figures 1 and 2).

The multi-dimensional version of the above scheme is given below.

$$\begin{aligned}
Y_{n+1}^k &= Y_{n+1}^k + (a^k - \frac{1}{2} \sum_{j=1}^m \Delta_x b_j \cdot b_j) h + \sum_{j=1}^m b^{k,j} \Delta W_{n+1}^j \\
&+ \sum_{j=1}^m \sum_{l=1}^m \Delta_x b_j \cdot b_l \cdot \Delta W_{n+1}^j W_{n+1}^l + \frac{1}{2} (3c^k - (c_c)^k) \Delta N_{n+1} \\
&+ \sum_{j=1}^m ((b_c)^{k,j} - b^{k,j}) \Delta W_{n+1}^j \Delta N_{n+1} + \frac{1}{2} ((c_c)^k - c^k) \Delta N_{n+1}^2 \\
&+ \sum_{j=1}^m (\Delta_x c^k b^{k,j} - (b_c)^{k,j} + b^{k,j}) \Delta Z_{n+1}^j, \tag{9}
\end{aligned}$$

where $b_i \cdot$ denotes the i -th column vector of b and $\Delta_x b_r \cdot$ is the matrix with ij -th element given by $\frac{\partial b_{rj}}{\partial x_i}$.

The figure below displays a Maple procedure which returns second order schemes of the form (9) for multi-dimensional jump-diffusion equations of the form (2).

Note that the syntax for the procedure ‘jump2MD’ is identical to that for the procedure ‘sceMD’.

```

jump2MD:=proc(a::array,b::array,c::array)
local k,h,f,g,z,temp,sol,i,u,j,solu,l,soln,sln,slon,ctmp,ctemp,btemp,
solut,soluto;
for k from 1 to coldim(b) do
h.k:=convert(col(b,k),matrix);
f:=(i,j)->diff(h.k[i,1],x[j]);
temp.k:=matrix(rowdim(h.k),rowdim(h.k),f);
sol.k:=(temp.k)*h.k;
for l from 1 to coldim(b) do
sln[k,l]:=((temp.k)*h.l)*(W.k[n+1])*(W.l[n+1]);
od; od;
solu:=evalm(sum('sol.m', 'm'=1..coldim(b)));
slon:=evalm(sum('sum('sln[m,r]', 'r'=1..coldim(b))', 'm'=1..coldim(b)));
for i to rowdim(c) do
ctmp.i:=[seq(x[j]=x[j]+c[i,1],j=1..rowdim(c))] od;
ctemp:=evalm(ctmp);
for i to rowdim(c) do
ctemp[i,1]:=subs(ctmp.i,ctemp[i,1]) od;
ctemp:=evalm(ctemp);
btemp:=evalm(b);
for i rowdim(b) do for j to coldim(b) do
btemp[i,j]:=subs(ctmp.i,btemp[i,j]) od od;
btemp:=evalm(btemp);
transpose(c)*b;
solut:=evalm(transpose(%));
g:=(i,j)->diff(solut[i,1],x[j]);
soluto:=matrix(rowdim(solut),rowdim(solut),g);
for i to rowdim(a) do
soln[i]:=Y.i[n+1]=Y.i[n]+(a[i,1]-(1/2)*solu[i,1])*h
+sum('b[i,j]*W.j[n+1]', 'j'=1..coldim(b))+slon[i,1]
+(1/2)*(3*c[i,1]-ctemp[i,1])*N[n+1]+sum('evalm(btemp-b)[i,j]*W.j[n+1]*N[n+1]',
'j'=1..coldim(b))+(1/2)*(ctemp[i,1]-c[i,1])*(N[n+1])2
+sum('evalm(soluto-btemp+b)[i,j]*Z.j[n+1]', 'j'=1..coldim(b));
for u to rowdim(a) do
soln[i]:=subs(x[u]=Y.u[n],soln[i]) od; od;
RETURN(eval(soln))
end:

```

By incorporating the jump times into the partition, simpler schemes, which are also of second order, may be constructed. The (second order) jump adapted version of equation (7) is given below.

$$Y_{n+1} = Y_n + \left(a - \frac{1}{2}bb_x\right)\Delta\tau_{n+1} + b\Delta W_{\tau_{n+1}} + \frac{1}{2}bb_x\Delta W_{\tau_{n+1}}^2 + c\Delta N_{\tau_{n+1}} + (b_c - b)\Delta W_{\tau_{n+1}}\Delta N_{\tau_{n+1}}, \quad (10)$$

where the new partition of $[t_0, T]$ is given by $0 \leq t_0 = \tau_0 < \tau_1 < \dots < \tau_N = T$, such that $\max_{0 \leq n \leq N}(\tau_{n+1} - \tau_n) \leq h$ w.p.1. Also, in equation (10) $Y_n = Y_{\tau_n}$, $\Delta\tau_{n+1} = \tau_{n+1} - \tau_n$, and $\Delta W_{\tau_{n+1}} = W_{\tau_{n+1}} - W_{\tau_n}$.

The corresponding Maple procedure for the above second order, jump adapted algorithm is given below in figure 5, and is appropriate for scalar jump-diffusion equations.

```
jumpadap:=proc(a::algebraic,b::algebraic,c::algebraic)
local soln;
soln:=Y[n+1]=Y[n]+(a-(1/2)*b*diff(b,x))*Delta*tau[n+1]
+b*W[tau[n+1]]+(1/2)*b*diff(b,x)*(W[tau[n+1]])^2+c*N[tau[n+1]]
+(subs(x=Y[n]+c,b)-b)*W[tau[n+1]]*N[tau[n+1]];
subs(x=Y[n],soln)
end;
```

Figure 5 Procedure ‘jumpadap’.

The syntax for the procedure ‘jumpadap’ is identical to that for the procedure ‘sce’ (see figures 1 and 2). On the same partition, a direct jump adapted scheme can be derived through noting that the solution path between jump times of the corresponding stochastic differential equation (of the form of equation (1)) restarts from the position immediately after the previous jump. The scheme, which is also of second order in the mean square sense, is given below.

$$Y_{n+1} = Y_n + c\Delta N_{\tau_{n+1}} + \left(\bar{a} - \frac{1}{2}\bar{b}\bar{b}_x\right)\Delta\tau_{n+1} + \bar{b}\Delta W_{\tau_{n+1}} + \frac{1}{2}\bar{b}\bar{b}_x\Delta W_{\tau_{n+1}}^2 \quad (11)$$

where the notation is the same as that in (10), and $\bar{a} = a(\tau_n, Y_n + c\Delta N_{\tau_{n+1}})$. The corresponding Maple procedure for the above second order, direct jump adapted algorithm is given below in figure 6, and is appropriate for scalar jump-diffusion equations.

```

directjmpad:=proc(a::algebraic,b::algebraic,c::algebraic)
local soln;
soln:=Y[n+1]=Y[n]+c*N[tau[n+1]]+(subs(x=Y[n]+c*N[tau[n+1]],a)
-1/2*sub(x=Y[n]+c*N[tau[n+1]],b)*diff(subs(x=x+c*N[tau[n+1]],b),x))*
Delta*tau[n+1]+subs(x=Y[n]+c*N[tau[n+1]],b)*W[tau[n+1]]
+1/2*subs(x=Y[n]+c*N[tau[n+1]],b)*diff(subs(x=x+c*N[tau[n+1]],b),x)*
(W[tau[n+1]])^2;
subs(x=Y[n],soln)
end:

```

Figure 6 Procedure ‘directjmpad’.

The syntax for the procedure ‘directjmpad’ is identical to that for the procedure ‘sce’ (see figures 1 and 2).

As Maghsoodi [13] notes, jump adapted schemes can be computationally inefficient, especially for nonlinear equations.

A Third Order Scheme

In practice, for a large class of jump-diffusion SDE, the scheme presented below is $O(h^3)$ in the mean square sense.

$$\begin{aligned}
Y_{n+1} = & Y_n + c_1 h + c_2 \Delta W + c_3 \Delta W^2 + c_4 \Delta N + c_5 \Delta W \Delta N \\
& + c_6 \Delta N^2 + c_7 \Delta Z + c_8 \Delta V + c_9 \Delta W^3 + c_{10} \Delta U \\
& + c_{11} h \Delta W + c_{12} \Delta N^3 + c_{13} \Delta W \Delta N^2 + c_{14} \Delta W \Delta N^3 \\
& + c_{15} h \Delta N + c_{16} h \Delta N^2 + c_{17} \Delta W^2 \Delta N + c_{18} \Delta Q \\
& + c_{19} \Delta W^2 \Delta N^2 + c_{20} \Delta W \Delta V + c_{21} \Delta T + c_{22} \Delta P \\
& + c_{23} \Delta N \Delta V + c_{24} \Delta N \Delta P,
\end{aligned} \tag{12}$$

where

$$\Delta Z = \int \Delta W_t dN_t, \quad \Delta V = \int \Delta N_t dW_t, \quad \Delta U = \int \Delta W_t dt, \quad \Delta Q = \int \Delta N_t dt,$$

$\Delta P = \int \Delta W_t^2 dN_t + \int \Delta N_t^2 dW_t$, $\Delta T = \int \Delta P_t dW_t$, $f = \int_{t_0}^{t_0+h}$, and

$$\begin{aligned}
c_1 &= a - \frac{1}{2}bb_x \\
c_2 &= b \\
c_3 &= \frac{1}{2}bb_x \\
c_4 &= \frac{1}{3}(c_{cc} - \frac{7}{2}c_c + \frac{11}{2}c) \\
c_5 &= \frac{1}{2}b_{cc} - \frac{1}{2}b + bc_x - bb_x - 2\alpha - 2\beta - 2\gamma + \rho + 2\theta \\
c_6 &= \frac{1}{2}(3c_c - 2c - c_{cc}) \\
c_7 &= -\frac{1}{2}b_{cc} + \frac{1}{2}b + 2\alpha + 2\beta + bb_x + 2\gamma - 2\theta - \frac{3}{2}\rho + \frac{1}{2}bc_x \\
c_9 &= \frac{1}{2}b(b_x^2 + bb_{xx}) \\
c_{10} &= ba_x - b_t - ab_x - \frac{1}{2}b^2b_{xx} \\
c_{11} &= b_t + ab_x - \frac{1}{2}b^2b_x \\
c_{12} &= \frac{1}{6}(c_{cc} - 2c_c + c) \\
c_{13} &= \frac{1}{2}(\rho - bc_x) \\
c_{15} &= c_t + ac_x + \frac{1}{2}bb_xc_x \\
c_{18} &= a_c - a - (c_t + ac_x + bb_x)c_x + \frac{1}{2}b^2c_{xx} - \alpha + \frac{1}{2}bb_x + \gamma \\
c_{19} &= \theta - \alpha - \frac{1}{2}bb_x \\
c_{20} &= 2\alpha - \theta \\
c_{22} &= -\frac{1}{2}b_{cc} + b_c - \frac{1}{2}b + \beta + 2\alpha + bb_x + 2\gamma - 2\theta - \frac{1}{2}bc_c - \frac{1}{2}\rho \\
c_{23} &= b_c - \frac{1}{2}b + 2\beta - \frac{1}{2}bc_x - \frac{3}{2}\rho - \frac{1}{2}b_{cc} + \gamma - \theta + \alpha + \frac{1}{2}bb_x \\
c_{24} &= -b_c + \frac{1}{2}b - \beta + \frac{1}{2}bc_x + \frac{1}{2}\rho + \frac{1}{2}b_{cc} - \gamma + \theta - \alpha - \frac{1}{2}bb_x \quad (13)
\end{aligned}$$

Here, $\alpha = \frac{1}{2}b_c(b_x)_c$, $\beta = b_c(c_x)_c$, $\gamma = \frac{1}{2}b(b_x c_x + b c_{xx})$, $\theta = b(b_x)_c(1 + c_x)$, $\rho = b(c_x)_c(1 + c_x)$, and the terms $c_8, c_{14}, c_{16}, c_{17}$, and c_{21} , which have larger variances, are set to zero. The notation in the above scheme is the same as in (7), with $b_{cc} = b(t_n, Y_n + c + c_c)$, and the functions are evaluated at the point (t_n, Y_n) .

The scheme above, in practice, is $O(h^2)$ in the mean square sense. However, it can be shown (Maghsoodi [13]) that for the class of jump-diffusion stochastic differential equation which satisfies $c_{10} \equiv 0$, the scheme is $O(h^3)$ in the mean square sense.

The figure below displays a Maple procedure which returns numerical schemes of the form (12) for scalar jump-diffusion equations.

```

jump3:=proc(a::algebraic,b::algebraic,c::algebraic)
local c1,c2,c3,c9,c10,c11,c4,c6,c12,c5,c7,c13,c15,c18,c19
c20,c22,c23,c24,alpha,beta,gamma,theta,rho,soln;
c1:=a-(1/2)*b*diff(b,x);
c2:=b;
c3:=(1/2)*b*diff(b,x);
c9:=1/2*b*((diff(b,x))^2+b*diff(b,x,x));
c10:=b*diff(a,x)-diff(b,t)-a*diff(b,x)-(1/2)*(b^2)*diff(b,x,x);
c11:=diff(b,t)+a*diff(b,x)-(1/2)*(b^2)*diff(b,x);
c4:=(1/3)*(subs(x=x+c+subs(x=x+c,c),c)-(7/2)*subs(x=x+c,c)+(11/2)*c);
c6:=(1/2)*(3*subs(x=x+c,c)-2*c-subst(x=x+c+subs(x=x+c,c),c));
c12:=(1/6)*(subs(x=x+c+subs(x=x+c,c),c)-2*subs(x=x+c,c)+c);
alpha:=(1/2)*subs(x=x+c,b)*subs(x=x+c,diff(b,x));
beta:=subs(x=x+c,b)*subs(x=x+c,diff(c,x));
gamma:=(1/2)*b*(diff(b,x)*diff(c,x)+b*diff(c,x,x));
theta:=b*subs(x=x+c,diff(b,x))*(1+diff(c,x));
rho:=b*subs(x=x+c,diff(c,x))*(1+diff(c,x));
c5:=(1/2)*subs(x=x+c+subs(x=x+c,c),b)-(1/2)*b+b*diff(c,x)-b*diff(b,x)
-2*alpha-2*beta-2*gamma+rho+2*theta;
c7:=-1/2*subs(x=x+c+subs(x=x+c,c),b)+(1/2)*b+2*alpha+2*beta
+b*diff(b,x)+2*gamma-2*theta-(3/2)*rho+(1/2)*b*diff(c,x);
c13:=(1/2)*(rho-b*diff(c,x));
c15:=diff(c,t)+a*diff(c,x)+(1/2)*b*diff(b,x)*diff(c,x);
c18:=subs(x=x+c,a)-a-(diff(c,t)+a*diff(c,x)+b*diff(b,x)*diff(c,x)
+(1/2)*(b^2)*diff(c,x,x))-alpha+(1/2)*b*diff(b,x)+gamma;
c19:=theta-alpha-(1/2)*b*diff(b,x);
c20:=2*alpha-theta;
c22:=-1/2*subs(x=x+c+subs(x=x+c,c),b)+subs(x=x+c,b)
-1/2*b+beta+2*alpha+b*diff(b,x)+2*gamma-2*theta-1/2*b*diff(c,x)
-1/2*rho;
c23:=subs(x=x+c,b)-1/2*b+2*beta-1/2*b*diff(c,x)-(3/2)*rho
-1/2*subs(x=x+c+subs(x=x+c,c),b)+gamma-theta+alpha
+(1/2)*b*diff(b,x);
c24:=-subs(x=x+c,b)+(1/2)*b-beta+(1/2)*b*diff(c,x)+(1/2)*rho
+(1/2)*subs(x=x+c+subs(x=x+c,c),b)-gamma+theta-alpha
-1/2*b*diff(b,x);
soln:=Y[n+1]=Y[n]+c1*h+c2*W[n+1]+c3*(W[n+1])^2+c4*N[n+1]
+c5*W[n+1]*N[n+1]+c6*(N[n+1])^2+c7*Z[n+1]+c9*(W[n+1])^3
+c10*U[n+1]+c11*h*W[n+1]+c12*(N[n+1])^3+c13*W[n+1]*(N[n+1])^2
+c15*h*N[n+1]+c18*Q[n+1]+c19*((W[n+1])^2)*(N[n+1])^2
+c20*(W[n+1])*(V[n+1])+c22*P[n+1]+c23*(N[n+1]*V[n+1])
+c24*(N[n+1]*P[n+1]);
subs(x=Y[n],soln)
end:

```

The syntax for the procedure ‘jump3’ is identical to that for the procedure ‘sce’ (see figures 1 and 2).

In order to implement the schemes given in this paper the stochastic integrals need to be evaluated. Often it is these integrals that make implementation difficult. Maghsoodi [13], [14],[15], Maghsoodi and Harris [16], and Kloeden and Platen [1] give formulas which are useful for digitally implementing such integrals. Burrage [17] has developed Maple routines which evaluate multiple stochastic Ito integrals.

Applications

Consider the jump-diffusion stochastic differential equations

$$dX_t = X_t dt + X_t dW_t + X_t dN_t \quad (14)$$

and

$$\begin{aligned} dX_t^1 &= X_t^1 dt + (X_t^2)^2 dW_t^1 + (X_t^1)^2 dW_t^2 + X_{t-}^2 dN_t \\ dX_t^2 &= X_t^2 dt + (X_t^1)^2 dW_t^1 + (X_t^2)^2 dW_t^2 + X_{t-}^1 dN_t. \end{aligned} \quad (15)$$

The figure below demonstrates the syntax required for the routines presented in this paper through construction of a second order scheme (equation (7)) for equation (14) and a first order scheme (equation (6)) for the system defined by equation (15).

```

>jump2(x,x,x);


$$Y_{n+1} = Y_n + \frac{1}{2}Y_n h + Y_n W_{n+1} + \frac{1}{2}Y_n W_{n+1}^2 + \frac{1}{2}Y_n N_{n+1} + Y_n W_{n+1} N_{n+1} + \frac{1}{2}Y_n N_{n+1}^2$$


>input1:=matrix(2,1,[x[1],x[2]]);


$$input1 := \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$


>input2:=matrix(2,2,[(x[2])^2,(x[1])^2,(x[1])^2,(x[2])^2]);


$$input2 := \begin{bmatrix} x_2^2 & x_1^2 \\ x_1^2 & x_2^2 \end{bmatrix}$$


>input3:=matrix(2,1,[x[2],x[1]]);


$$input3 := \begin{bmatrix} x_2 \\ x_1 \end{bmatrix}$$


>sceMD(input1,input2,input3);

table([
1 = (Y1n+1 = Y1n + Y1nh + Y2n2W1n+1 + Y1n2W2n+1 + Y2nNn+1)
2 = (Y2n+1 = Y2n + Y2nh + Y1n2W1n+1 + Y2n2W2n+1 + Y1nNn+1)
])

```

Figure 7 Procedures ‘jump2’ and ‘sceMD’ applied to equations (14) and (15), respectively.

Note that the output in figure 7 employs the convention that Yk_n is the approximate solution to X_t^k

Discussion

Numerical schemes which are $O(h)$, $O(h^2)$, and $O(h^3)$ in the mean square sense have been presented in this paper, including two jump adapted schemes which are $O(h^2)$. These schemes were derived by Maghsoodi [13]. The routines presented here are Maple implementations of these schemes. Such automation allows any jump-diffusion stochastic differential equation to be entered as input. The resulting output is then the constructed scheme of appro-

priate order. The use of the symbolic manipulator Maple in this area is useful due to the amount of algebra involved, particularly for multi-dimensional equations and higher order schemes.

Possible extensions to the programs presented here could be the inclusion of multi-dimensional versions of the jump adapted schemes (equations (10) and (11)) and of the third order scheme (equation (12)).

The programs presented in this paper together with the ‘stochastic’ package (Cyganowski [6]) can be downloaded from the following web site:

<http://www3.cm.deakin.edu.au/sash/maple.html>

References

- [1] P.E. Kloeden and E. Platen, *Numerical Solution of Stochastic Differential Equations*, Springer-Verlag, Berlin Heidelberg, 1992.
- [2] I.I. Gikhman and A.V. Skorokhod, *Stochastic Differential Equations*, Springer-Verlag, Berlin, 1972.
- [3] B. Grigelionis, *Random point processes and martingales*, Lietuvos Matematikos Rinkiny, 15 (1975), pp. 101-114.
- [4] K. Ito, *On stochastic differential equations*, Mem. Am. Math. Soc., 4 (1951), pp. 1-51.
- [5] G.N. Milshstein, *Approximate integration of stochastic differential equations*, Theor. Prob. Applics., 19 (1974), pp. 557-562.
- [6] S.O. Cyganowski, *A MAPLE Package for Stochastic Differential Equations*, in Computational Techniques and Applications: CTAC95, World Scientific, Singapore, 1996, pp. 223-230.
- [7] S.O. Cyganowski, *Solving Stochastic Differential Equations with Maple*, MapleTech, 3 (1996), pp. 38-40.
- [8] S.O. Cyganowski and P.E. Kloeden, *Stochastic Stability Examined through Maple*, in 15th IMACS World Congress on Scientific Computation, Modelling and Applied Mathematics, Vol. 1, Wissenschaft and Technik Verlag, Berlin, 1997, pp. 437-442.

- [9] P.E. Kloeden and W.D. Scott, *Construction of Stochastic Numerical Schemes through Maple*, MapleTech, 10 (1993), pp. 60-65.
- [10] K. Xu, *Stochastic pitchfork bifurcation: numerical simulations and symbolic calculations using MAPLE*, Mathematics and Computers in Simulation, 38 (1995), pp. 199-207.
- [11] W.S. Kendall, *Computer algebra and stochastic calculus*, Notices Amer. Math. Soc., 37 (1990), pp. 1254-1256.
- [12] E. Valkeila, *Computer algebra and stochastic analysis, some possibilities*, CWI Quarterly, 4 (1991), pp. 229-238.
- [13] Y. Maghsoodi, *Mean Square Efficient Numerical Solution of Jump-Diffusion Stochastic Differential Equations*, Indian J. Statistics, 58 (1996), pp. 25-47.
- [14] Y. Maghsoodi, *On approximate integration of a class of stochastic differential equations*, D. Phil. Thesis, University of Oxford, (1983).
- [15] Y. Maghsoodi, *Taylor expansion of semi-group operators and efficient discretization and simulation of jump-diffusion stochastic differential equations*, Working Paper O.R. 50A, Dept. Math., University of Southampton, 1992.
- [16] Y. Maghsoodi and C.J. Harris, *In-probability approximation and simulation of non-linear jump-diffusion stochastic differential equations*, IMA Journal of Mathematical Control and Information, 4 (1987), pp. 65-92.
- [17] P. Burrage, *Runge-Kutta Methods for Stochastic Differential Equations*, D. Phil. Thesis, University of Queensland, (1999).