Computational Methods in Finance, Lecture 2, Diffusions and Diffusion Equations.

Jonathan Goodman *
Courant Institute of Mathematical Sciences, NYU

September 24, 1999

1 Introduction

This lecture and the next are about finite difference methods for solving the diffusion equations that arise in financial applications. Before getting to finite differences, we review stochastic differential equations. As in Lecture 1, we discuss the forward and backward equations and the differences between them.

2 Diffusions and Diffusion Equations

Lecture 1 discusses Markov processes in discrete state space and discrete time. Now we turn to continuous time and continuous state space. The state at time t is a vector, $X \in \mathbf{R}^n$ consisting of n components, or "factors", $X = (X_1, \ldots, X_n)$. The dynamics are given by the Ito differential equation

$$dX(t) = a(X(t))dt + b(X(t))dZ \ . \tag{1}$$

Here $Z(t)$ is a vector of m independent standard Brownian motions. For each x, there is a "drift", $a(x)$, and an $n \times m$ matrix $b(x)$, that is related

*goodman@cims.nyu.edu, or http://www.math.nyu.edu/faculty/goodman,

to the volatility. There is no reason that m, the number of noise sources, should equal n, the number of factors, but there is no reason ever to have more noises than factors. The columns of b are vectors in $\mathbf{R}^n$. Column j gives the influence of noise j on the dynamics. If these columns do not span $\mathbf{R}^n$, then the diffusion is "degenerate". Otherwise, it is nondegenerate. Both types come up in financial applications.

As in Lecture 1, there are forward and backward evolution equations that are dual to each other. The forward equation is for $u(x,t)$, the probability density for $X(t)$. This is the "diffusion equation"

$$\partial_t u = -\sum_{j=1}^{n} \left(\partial_{x_j} a_j(x) u \right) + \frac{1}{2} \sum_{j,k=1}^{n} \partial_{x_j} \partial_{x_k} \left(\mu_{jk}(x) u \right) \quad . \tag{2}$$

The matrix of diffusion coefficients, μ, is related to b by

$$\mu(x) = b(x) \cdot b^*(x) \quad . \tag{3}$$

We write M^* for the transpose of a matrix, M. The coefficients, $a_j(x)$, in (2) are the components of a in (1).

A strict mathematical derivation of (2) from (1) or vice versa is outside the scope of this course. However, some aspects of (2) are natural. First, because u is a probability density, the integral of $u(x,t)$ over x should be independent of t. That will happen if all the terms on the right of (2) are derivatives of something (i.e. $\partial_x(a(x)u)$ rather than $a(x)\partial_x u$). This is sometimes called "conservation form". The second term on the right of (2) involves two derivatives. Someone used to physical modeling might expect it to take the form

$$\text{WRONG} \qquad \frac{1}{2} \sum_{j,k=1}^{n} \partial_{x_j} \mu_{jk}(x) \partial_{x_k} u \quad . \qquad \text{WRONG} \tag{4}$$

The actual form (2) has the "martingale" property that, if there is no drift ($a \equiv 0$), then the expected value of X does not change with time. To see this, use (2) with $a = 0$ and compute

$$\begin{aligned} \partial_t \mathbf{E}\left[X(t)\right] &= \partial_t \int x u(x,t) dx \\ &= \int x \frac{1}{2} \sum_{j,k=1}^{n} \partial_{x_j} \partial_{x_k} \mu_{jk}(x) u d \\ &= 0 \quad . \end{aligned}$$

The last line follow from the one above if you integrate by parts twice to put the two derivatives on the x. The result would generally not be zero using (4) instead of (2).

Here is an equally rigorous derivation of the relation (3) between volatility and diffusion coefficients. Suppose first that $n = m = 1$. The number μ should depend on b in some way. Observe that the diffusion governed by (1) will be unchanged of b is replaced by $-b$, because $Z(t)$ is indistinguishable from $-Z(t)$. This suggests the formula $\mu = b^2$. In general, we need a matrix analogue of $\mu = b^2$ that produces an $n \times n$ matrix, μ, from an $n \times m$ matrix, b. The simplest possibility is (3).

As another check, suppose b is constant b and $a = 0$. Then we can match moments. Use (1) and assume $X(0) = 0$ and[1] $Z(0) = 0$ and get $X(t) = bZ(t)$. From this it follows that

$$\text{cov}(X) = \mathbf{E}\left(X(t)X^*(t)\right) = bb^*t \quad .$$

On the other hand, from (2) we compute

$$\begin{aligned} \partial_t \mathbf{E}\left(XX^*\right) &= \partial_t \int xx^* u(x,t)dx \\ &= \frac{1}{2}\int xx^* \sum_{jk} \partial_{x_j}\partial_{x_k}\mu_{jk}u dx \\ &= \mu \quad . \end{aligned}$$

which again agrees with (3).

The drift term in (2), $\partial_x au$, corresponds to the drift term in (1), $a(X)dt$. It is easy to see what the term would be if a were constant (independent of x and t) and b were zero. In that case the solution of (1) would be $X(t) = X(0) + at$. For this reason, the probability density is also simply shifted with speed a: $u(x,t) = u(x - at, 0)$. This function satisfies (2) (if the signs are right).

As in Lecture 1, the simplest backward equation is for the expected payout starting at x at time t:

$$f(x,t) = \mathbf{E}\left[f_T(X(T))|X(t) = x\right] \quad .$$

More complicated expectations satisfy more complicated but related equations. The backward equation for f is

$$\partial_t f + \sum_j a_j(x)\partial_{x_j} f + \frac{1}{2}\sum_{jk}\mu_{jk}(x)\partial_{x_j}\partial_{x_k} f = 0 \quad . \tag{5}$$

[1]The Wiener process is usually defined to that $Z(0) = 0$.

Still following Lecture 1, this is supplemented with "initial data" given at the final time, T, $f(x,T) = f_T(x)$, and determines $f(x,t)$ for $t < T$. Again, we can express unconditional expectation in terms of conditional expectation starting from time t and the probability density for $X(t)$:

$$\mathbf{E}\left[f_T(X(T))\right] = \int f(x,t)u(x,t)dx \quad . \tag{6}$$

The fact that the right side of (6) is independent of t allows us to derive (5) from (2) or vice versa. Finally, f satisfies a "maximum principle":

$$\min_y f(y,T) \le f(x,t) \le \max_y f(y,T) \quad \text{if } t < T.$$

The probability interpretation of f makes this obvious; the expected reward cannot be less than the least possible reward nor larger than the largest.

We can motivate (5) using a PDE (Partial Differential Equation) version of the argument from lecture 1. For now, we will just give a way in which (5) and (2) are consistent with each other. The more detailed derivation of (5) from (2) will have to wait until we know more about Brownian motion. For now, we just compute the expected value, at time 0, of the payout at time T by "averaging" over the possible states at some intermediate time, t. Using the probability density for $X(t)$, this gives

$$\begin{aligned} \mathbf{E}\left[f_T(X(t))\right] &= \int \mathbf{E}\left[f_T(X(T)) \mid X(t) = x\right] u(x,t)dx \\ &= \int f(x,t)u(x,t)dx \quad . \end{aligned}$$

Since the left side does not depend on t, the right side also must be independent of t. This leads to

$$\begin{aligned} 0 &= \frac{d}{dt}\int f(x,t)u(x,t)dx \\ &= \int \left\{(\partial_t f(x,t))u(x,t) + f(x,t)\partial_t u(x,t)\right\} dx \quad . \end{aligned}$$

The evolution equation (2) transforms this into

$$\begin{aligned} 0 = \int \Big\{ &(\partial_t f(x,t))\, u(x,t) \quad - \quad f(x,t)\sum_{j=1}^{n} \partial_{x_j}\left(a_j(x)u(x,t)\right) \\ &+ \quad f(x,t)\frac{1}{2}\sum_{jk} \partial_{x_j}\partial_{x_k}\left(\mu_{jk}(x)u(x,t)\right)\Big\} dx \quad . \end{aligned}$$

Now we want to integrate by parts. Normally we would get terms with the derivatives on f and "boundary" terms. Here, there are no boundary terms and[2] $u \to 0$ as $x \to \infty$. Upon integrating by parts and grouping terms, we come to:

$$0 = \int \left(\partial_t f + \sum_j a_j(x) \partial_{x_j} f + \frac{1}{2} \sum_{jk} \mu_{jk}(x) \partial_{x_j} \partial_{x_k} f \right) u(x,t) dx \quad .$$

The simplest way for this integral to be zero automatically is for f to satisfy (5).

2.1 Abstract duality

The relation between (2) and (5) may be stated more abstractly using the language of adjoint operators. This has the advantage of clarifying the relationship between the continuous time, continuous X version we are discussing here and the discrete time and discrete X version from lecture 1. Last lecture, we distinguished between row and column vectors. The abstract version of this distinction is the distinction between a vector space, V, and its "dual space", V^*. For example, if V is the space of row vectors, then V^* is the space of column vectors. The abstract relationship between a vector space and its dual is that *elements of V^* are linear functionals on V*. A linear functional is a linear function that produces a number from a vector. In the row and column vector setting, a column vector, f produces a function on row vectors by taking row vector u to the matrix product $u \cdot f$. Any such abstract pairing is written[3] (u, f). Conversely, any such linear functional corresponds to a column vector. Summarizing, the spaces of row vectors and column vectors of a given dimension have a natural "duality pairing" given by the matrix product: $(u, f) = u \cdot f$. The matrix product $u \cdot f$ is the definition of (u, f) in this case.

In our current setting of functions of a continuous variable (or set of variables), x, the "duality relation" is defined by integration. If $f(x)$ is a (payout) function, then we may define a linear functional on (probability)

[2]For example, if a and b are bounded it is impossible for $X(t)$ to "escape to ∞" in finite time. This implies that $u(x)$ goes to zero as x goes to infinity.

[3]Notations of this kind were first introduced by the English physicist Dirac. He would have written $< u \mid f >$. The left part, $< u \mid$, would have been called a "bra" vector, and the right, $\mid f >$, a "ket". Putting them together forms the Dirac"bracket".

functions by taking a function $u(x)$ to $\int u(x)f(x)dx$. That is, the duality relation given by

$$(u, f) = \int u(x)f(x)dx \quad .$$

Here again, the right side is the definition of the left side. I wish it were possible to think of functions u as infinite continuous rows and functions f as corresponding columns.

The general abstract analogue of a square matrix is a "linear operator", which is a map from V to V, or from V^* to V^*. Now suppose we have a dual pair of vector spaces and a linear operator, A, on V, then there is a "dual" operator on V^*. The dual of A is written A^*. If $f \in V^*$, then A^*f is another element of V^*. The duality relation for operators is that $(Au, f) = (u, A^*f)$ for every $u \in V$ and $f \in V^*$.

In the case of row and column vectors from lecture 1, a matrix, M, acts as a linear operator on row vectors by matrix multiplication from the right. That is Au is the row vector given by the matrix product $u \cdot M$. You should not think of A as a matrix, or Au as matrix vector multiplication, because u has the wrong shape multiplication from the left be a matrix. The dual of A is also given by the matrix M, this time acting on *column* vectors, f by matrix multiplication from the left. That is, A^*f is given by the matrix product $M \cdot f$, which is another column vector. The duality relation, $(Au, f) = (u, A^*f)$, in this case boils down to the associativity of matrix multiplication. First, $(Au, f) = (u \cdot M, f) = (u \cdot M) \cdot f$, also $(u, A^*f) = (u, M \cdot f) = u \cdot (M \cdot f)$. Because matrix multiplication is associative, $(u \cdot M) \cdot f = u \cdot (M \cdot f)$. Note, in this last formula, the parentheses refer to groupings in matrix multiplication rather than the duality pairing. This is a flaw in accepted mathematical notation that I am powerless to correct.

Now we come to the point of this subsection, the duality relation connecting the equations (2) and (5). If $u = u(x)$ is a function of the continuous variable x, then we can define the linear operator that gives the function $v(x)$ by

$$v(x) = -\sum_j^n \partial_{x_j} a_j(x)u(x) + \frac{1}{2}\sum_{jk} \partial_{x_j}\partial_{x_k}\mu_{jk}(x)u(x) \quad .$$

In operator notation, we might write

$$(Au)(x) \text{ or } Au(x) = -\sum_j \partial_{x_j} a_j(x)u(x) + \frac{1}{2}\sum_{jk} \partial_{x_j}\partial_{x_k}\mu_{jk}(x)u(x) \quad .$$

The evolution equation (2) is then $\partial_t u = Au$.

We find the dual operator for A through the definition of the duality pairing and integration by parts. Using the notation $g = A^* f$, we have, using integration by parts,

$$\begin{aligned}
(Au, f) &= (u, A^* f) = (u, g) \\
\int Au(x) f(x) dx &= (u, g) \\
\int \left\{ -\sum_j \partial_{x_j} a_j(x) u(x) + \frac{1}{2} \sum_{jk} \partial_{x_j} \partial_{x_k} \mu_{jk}(x) u(x) \right\} f(x) dx &= (u, g) \\
\int u(x) \left\{ \sum_j a_j(x) \partial_{x_j} f(x) + \frac{1}{2} \mu_{jk}(x) \sum_{jk} \partial_{x_j} \partial_{x_k} f(x) \right\} &= \int u(x) g(x) dx \quad .
\end{aligned}$$

Look at the last line here. If we want this to be true for every function $u(x)$, we should set the left parts of each side equal. That is

$$g(x) = A^* f(x) = \sum_j a_j(x) \partial_{x_j} f(x) + \frac{1}{2} \mu_{jk}(x) \sum_{jk} \partial_{x_j} \partial_{x_k} f(x) \quad .$$

The backward evolution equation (5) may now be written

$$\partial_t f = -A^* f \quad .$$

The derivation of (2) from (5) may be written abstractly too. For each t we have a vector u, which we write $u(t)$. Be careful this $u(t)$ is a vector function of t rather than an "ordinary" function. That is, $u(t)$ is an element of V rather than being a single number. We similarly define $f(t) \in V^*$. The expectation that is independent of t is $u(t), f(t) = \int u(x,t) f(x,t) dx$. Differentiating and using the abstract form of the u evolution equation, we get[4]

$$0 = \partial_t (u(t), f(t)) = (\partial_t u(t), f(t)) + (u(t), \partial_t f(t)) \quad .$$

Using $\partial_t u = Au$ and taking the dual, this gives

$$(u(t), A^* f(t)) = (u(t), -\partial_t f(t)) \quad .$$

The simplest way to make this true is to have $\partial_t f(t) = -A^* f(t)$. This is the abstract form of the f evolution equation (5).

[4]Let's believe that the product rule for differentiation works for duality pairings as it does for ordinary products, which it does.

2.2 Boundaries and boundary conditions

In addition to initial conditions, diffusion equations often come with boundary conditions. In this case, the "domain" will be a subset of all possible x values. The boundary of the domain will be called B. Financial applications often give rise to diffusion equations without boundary conditions. For example, there are no natural restrictions on the price of a stock. Boundary conditions do come in, for example, when working with knockout features. A knockout at a set B means that there will be no payout if $X(t) \in B$ for any $0 \leq t \leq T$. To value such an instrument, we use the backwards equation with "Dirichlet" boundary condition $f(x,t) = 0$ for $x \in B$. If the forward equation is applicable, we also apply the Dirichlet boundary condition to u. In the latter case, we usually have $\int u dx < 1$. This $u(x,t)$ represents the probability density for those paths that have never touched the boundary. The complimentary probability is the probability of touching the boundary at some time:

$$\int u(x,t)dx + \mathbf{Pr}\left(X(t') \in B \text{ for some } t' \leq T\right) = 1 \quad .$$

Most diffusions in finance do not "live" in all of R^n, but in a natural subset. For example, stock prices and interest rates are usually positive. In these cases, the diffusion coefficients may go to zero, as X gets close to the edge, in such a way that $X(t)$ can never leave the set. For example, in the model used by Black and Scholes, $dS = rSdt + \sigma SdZ$, the $S(t)$ can never become negative if it starts positive. In these cases, no extra boundary conditions need to be specified. We do not give a boundary condition at $S = 0$ when solving the Black Scholes equation.

Often the initial data, $f(x,T)$, or $u(x,0)$, are singular. A singularity is an x value where the function is not smooth. For example, if $X(0) = x_0$ is known, then the initial probability density is a delta function: $u(x,0) = \delta(x - x_0)$. The payout for a stock option has a jump in its derivative at the strike price. If the diffusion is nondegenerate (the coefficient matrix, μ, is positive definite), such singularities quickly smooth out. This may or may not be true for degenerate diffusions.

3 The Heat Equation, a Model Problem

In designing and understanding computational methods, we try to identify model problems. A model problem is a simpler computation that helps us

focus on the essential difficulties while temporarily avoiding other complexities of our actual problem. We must keep in mind that a model problem can be an oversimplification, some of its features are not shared by the actual problem.

Our first model problem for diffusion equations will be the "heat equation"

$$\partial_t u = \frac{1}{2}\partial_x^2 u \tag{7}$$

in one space dimension. This is to be solved together with initial data $u(x,0) = u_0(x)$. This problem has in common with general diffusion equations (2) or (3) the smoothing property, time step constraints for explicit difference methods, and infinite propagation speed with limited spreading. It has the unusual features that the forward and backward equations are (up to a sign) the same, and that there is a simple algebraic formula for the fundamental solution.

4 Finite differences and marching methods for the heat equation

Finite difference, or "marching methods", compute an approximation to u on a discrete grid (or lattice or mesh). In the simplest case, the grid is defined by a small space step, Δx, and time step, Δt. The mesh ponts, uniformly spaced, are defined by $x_k = k \cdot \Delta x$ and $t_n = n \cdot \Delta t$. The approximate solution values we are calculating will be called U_k^n. The accuracy of the approximation is determined by the difference between U_k^n and the exact values $u(x_k, t_n)$. For example, the forward Euler method is usually "second order accurate" in space and first order in time, which means that[5]

$$U_k^n - u(x_k, t_n) = O(\Delta x^2 + \Delta t) \quad .$$

In marching methods, the numbers U_k^n are computed one "time step" at a time. First, the numbers $U_k^0 = u(x_k, 0)$ are taken from the given initial data. Now suppose we have computed U_k^n for all k. We want to compute U_k^{n+1} for all k. This is a time step. Note that we do not compute U_k^{n+1} from

[5]For definitions of order of accuracy, see my lecture notes on Scientific Computing, or a good book on numerical analysis, such as that by Dahlquist and Björk, or that of Isaacson and Keller.

U_k^n alone. In the simplest case, again forward Euler, U_k^{n+1} depends on U_{k-1}^n, U_k^n, and U_{k+1}^n.

The main idea is to replace each of the partial derivatives in (7) with a finite difference approximation to it. The finite difference approximations should be chosen so that it is possible to compute the U_k^{n+1} from the U_k^n, and so that the overall method is stable. Stability is a real worry, but one that we will face up to later.

To implement this idea, we want approximations for the partial derivatives $\partial_t u$ and $\partial_x^2 u$ at the grid point (x_k, t_n). We will approximate $\partial_t u$ by the "forward difference":

$$\begin{aligned} (\partial_t u)(x_k, t_n) &\approx \frac{u(x_k, t_n + \Delta t) - u(x_k, t_n)}{\Delta t} \\ &\approx \frac{u(x_k, t_{n+1}) - u(x_k, t_n)}{\Delta t} \\ \partial_t u(x_k, t_n) &\approx \frac{U_k^{n+1} - U_k^n}{\Delta t} \quad . \end{aligned} \tag{8}$$

For $\partial_x^2 u$, we will use a central second difference

$$\begin{aligned} \left(\partial_x^2 u\right)(x_k, t_n) &\approx \frac{u(x_k + \Delta x, t_n) - 2u(x_k, t_n) + u(x_k - \Delta x, t_n)}{\Delta x^2} \\ &\approx \frac{u(x_{k+1}, t_n) - 2u(x_k, t_n) + u(x_{k-1}, t_n)}{\Delta x^2} \\ \partial_x^2 u(x_k, t_n) &\approx \frac{U_{k+1}^n - 2U_k^n + U_{k-1}^n}{\Delta x^2} \quad . \end{aligned} \tag{9}$$

While the formulae (8) and (9) are approximations, we use them to make an exact definition of U_k^n. That is, we choose the U_k^n to satisfy

$$\frac{U_k^{n+1} - U_k^n}{\Delta t} = \frac{1}{2} \frac{u(x_{k+1}, t_n) - 2u(x_k, t_n) + u(x_{k-1}, t_n)}{\Delta x^2} \quad . \tag{10}$$

A little manipulation of (10) leads to

$$U_k^{n+1} = a U_{k-1}^n + b U_k^n + c U_{k+1}^n \quad , \tag{11}$$

where

$$a = \frac{\Delta t}{2\Delta x^2} \quad , \quad b = 1 - \frac{\Delta t}{\Delta x^2} \quad . \quad c = \frac{\Delta t}{2\Delta x^2} \quad . \tag{12}$$

It is now clear, we can use (11) to compute all the U_k^{n+1} if we already know all the U_k^n. As I said earlier, each value at the future time depends on

three values at the present time. The coefficients (12) have a probabilistic interpretation that we will discuss later when we show that this is exactly the "trinomial tree" method.

For future reference, note that the coefficients a, b, and c, are all positive only if $\Delta t < \Delta x^2$. In practice, if you violate this constraint, the computation will be "unstable". In fact, the computation will quickly "blow up". Later notes will discuss this in detail.

I close with a note about programming. The natural way to program the formula (11) in C/C++ would be to use an array with two indices. However, there is an equally efficient way to program it that uses only two one dimensional arrays. Since we typically need to use a large number of time steps, this is a large saving in storage. If we call the values on the right of (11) just U_k and the values on the left V_k, then we just compute the values in the V array from the values in the U array. Having done this, we reverse the roles of U and V, that is, overwrite the U array with values computed from V. The code to take two time steps might go

```
timeStep(V,U);
timeStep(U,V);
```

where `timeStep` is the routine that applies the formula (11) for all k values.