Errata and comments for *Numerical Methods in Finance and Economics* (2nd edition)

Paolo Brandimarte

November 22, 2011

Typos and errors

Page 68, last displayed equation. $\mathrm{E}[U(X)]$ should just read $U(X)$ (which is indeed defined as $\mathrm{E}[u(X)]$.

Page 109, second displayed equation. $F(S(T), T) = \max\{S(T)-K, 0\}$ should read $f(S(T), T) = \max\{S(T) - K, 0\}$.

Page 109, line -4. "This equation applies to any option whose payoff depends only on the current price of the underlying asset, or its price at maturity."

should read

"This equation applies to any option whose payoff depends only on the current price of the underlying asset and its time to maturity."

Page 109, lines 17–18. There is a mixup between "1" and "∞". The text should read

- $\|\mathbf{x}\|_\infty \equiv \max_{1\le i\le n} |x_i|$, which is known as L_∞ norm;
- $\|\mathbf{x}\|_1 \equiv \sum_{i=1}^n |x_i|$, which is known as L_1 norm.

Page 109, last line. The dot closing the line

```
sum(abs(v).^p)^(1/p).
```

may be confusing and should be deleted.

Page 169, second displayed equation. The expression

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \omega\mathbf{r}^{(k)} = \omega\mathbf{x}^{(k+1)} + (1-\omega)\mathbf{x}^{(k)}$$

is a bit confusing if regarded as an equation rather than an assignment of variables within an algorithm. It is better to write

$$\hat{\mathbf{x}}^{(k+1)} = \mathbf{x}^{(k)} + \omega\mathbf{r}^{(k)} = \omega\mathbf{x}^{(k+1)} + (1-\omega)\mathbf{x}^{(k)}$$

as in Eq. (3.6).

Page 181, MATLAB code in Fig. 3.16. The lines

```
peq = polyfit(EquiNodes,runge(EquiNodes),10);
```

and

```
plot(x,polyval(p10,x));
```

are not consistent. Replace the second one with `plot(x,polyval(peq,x));`

Page 196, line 8. "... implicit volatility" should read "... implied volatility."

Page 197, lines 11–14. The claim that the MATLAB function `blsimpv` uses Newton's method is not true anymore, as the function has been changed with respect to previous versions. In fact, when starting from very large values of volatility, Newton's method may produce a negative value of volatility, which leads to an error in applying Black and Scholes formulae. This is an instance of a general weakness of this method, which is not globally convergent.

Page 226, section 4.3.1. A general comment is in order concerning statements like `rand(10,1)`, `rand('seed',0)`, and `rand('state',0)`. This way of generating random numbers is a bit obsolete now, as MATLAB is getting more and more object oriented. Now a class of random generators is offered: you instantiate objects of the class and call their methods.

Page 233, MATLAB code in Fig. 4.9. The function can be streamlined a bit and the line

```
loc=sum(rand*cumprobs(N) > cumprobs) + 1;
```

can be changed to

```
loc=sum(rand > cumprobs) + 1;
```

In doing so, we rely on the fact that the probabilities supplied as input arguments are correct and do add up to 1.

Page 236, lines 2 and 10. Add a dot after `sigma` in line 2, and change "method" to "methods" in line 10.

Page 241, lines 2 and 6. Line 2 should read "... to probability level $1 - \alpha/2$; by the same token $z_{1-\alpha}$ in line 6 should read $z_{1-\alpha/2}$.

Page 247, MATLAB code in Fig. 4.16. The input parameter `NRepl` should be replaced by `NPairs` (which is how it is referred to in line -4). Indeed, that parameter defines the number of pairs; the number of observations is twice that much.

Page 249, caption in Fig. 4.17. To be more consistent in notation, the strikes X_1, X_2, and X_3 should be replaced by K_1, K_2, and K_3, respectively.

Page 254, line 1. Since we are talking about European-style options, "expiration" should be replaced by "maturity."

Page 267, fourth displayed equation. Change

$$e^{-\left[2(\alpha-\beta)Y-\alpha^2+\beta^2\right]/2\xi^2}$$

to

$$e^{\left[2(\alpha-\beta)Y-\alpha^2+\beta^2\right]/2\xi^2}$$

Page 268, MATLAB code in Fig. 4.28. There are several errors in this code (sorry...):

- When preallocating `MCError` and `MCISError`, I should use 100 and not `NRepl`, where 100 is the number of experiments tried, each consisting of a Monte Carlo simulation with `NRepl` replications.
- When calling `blsprice`, `BlsMC2`, and `BlsMCIS` the order of input arguments is wrong, as time to maturity and volatility should be swapped.
- Within the iterations the preallocated arrays `MCError` and `MCISError` are erased and replaced by a scalar, as they are not indexed by `k`.

The correct code should read something like:

```
% CheckBlsMCIS.m
S0 = 50;
K = 80;
r = 0.05;
sigma = 0.4;
T = 5/12;
NExp = 100; % number of Monte Carlo runs
NRepl = 100000; % number of replications within each run
MCError = zeros(NExp,1);
MCISError = zeros(NExp,1);
TruePrice = blsprice(S0,K,r,T,sigma);
randn('state',0);
for k=1:NExp
    MCPrice = BlsMC2(S0,K,r,T,sigma,NRepl);
    MCError(k) = abs(MCPrice - TruePrice)/TruePrice;
end
randn('state',0);
for k=1:NExp
    MCISPrice = BlsMCIS(S0,K,r,T,sigma,NRepl);
    MCISError(k) = abs(MCISPrice - TruePrice)/TruePrice;
end
fprintf(1,'Average Percentage Error:\n');
fprintf(1,' MC    = %6.3f%%\n', 100*mean(MCError));
fprintf(1,' MC+IS = %6.3f%%\n', 100*mean(MCISError));
```

Page 298, MATLAB snapshot at the bottom of page. The call

```
>> sol = transport(xmin, dx, xmax, dt, tmax, c, 'f0transp');
```

is obsolete. Now, rather than using a string containing the name of the function, the function handle is used, like

```
>> sol = transport(xmin, dx, xmax, dt, tmax, c, @f0transp);
```

Page 307, line -3. Replace the second boundary condition

$$\phi_{1,j} = f_N(j\,\delta t) = f_{Nj}$$

with

$$\phi_{N,j} = f_N(j\,\delta t) = f_{Nj}$$

Page 310, line -4. Replace $\delta x = 0.1$ with $\delta x = 0.01$; the same applies to the caption of Fig. 5.18.

Page 313, first displayed equation. Replace $\ldots + O(\delta x^2)$ with $\ldots + O(\delta t^2)$.

Page 314, displayed equation in the middle of page. There is a ρ missing. Replace the equation with

$$\alpha_k = \frac{2 - 4\rho\,\sin^2\left(\frac{k\pi}{2N}\right)}{2 + 4\rho\,\sin^2\left(\frac{k\pi}{2N}\right)}, \qquad k = 1, 2, \ldots, N-1.$$

Page 315, displayed equation after Eq. (5.26). The second constant should read

$$\rho_y = \frac{\delta t}{(\delta y)^2}$$

Page 371, second displayed equation. The equation should read:

$$\mathbf{c}'\mathbf{x} = \mathbf{c}'_B\mathbf{x}_B + \mathbf{c}'_N\mathbf{x}_N = \mathbf{c}'_B(\hat{\mathbf{b}} - \mathbf{A}_B^{-1}\mathbf{A}_N\mathbf{x}_N) + \mathbf{c}'_N\mathbf{x}_N = \hat{f} + \hat{\mathbf{c}}'_N\mathbf{x}_N$$

Page 403, Fig. 7.2. The figure is not quite consistent, as the middle node in the third layer should be Sud. If we choose $ud = 1$, clearly we may write S rather than Sud, but then the whole figure should be changed.

Page 403, line 5. Replace "form" with "from."

Page 403, Section 7.1.1. A general comment is in order, as the notation may be somewhat sloppy. Strictly speaking, everything we say here is conditional on the value S_t. We could write expected values and variances conditional on the event $S_t = s_t$, distinguishing the random variable from its realized value. However, the message should be clear enough.

Page 417, last displayed equation. The equations for the log prices should read:

$$dX_1 = \nu_1 dt + \sigma_1\, dW_1$$
$$dX_2 = \nu_2 dt + \sigma_2\, dW_2$$

Page 418, last displayed equation. In the four probabilities, replace every occurrence of μ_1 and μ_2 with ν_1 and ν_2, respectively.

Page 422, Eq. (7.7). The three equations seem to yield p_u three times; of course, the three probabilities are p_u, p_m, and p_d, respectively.

Page 423, line 3. Replace $\delta x = 3\sqrt{\delta t}$ with $\delta x = \sigma\sqrt{\delta t}$.

Page 460, MATLAB code in Fig. 8.19. There is an inconsistency between `NormMat` and `RandMat`; the second variable should be replaces (twice) by the first one.

Page 470, MATLAB code in Fig. 8.27. The code is completely wrong, since it is the same as in Fig. 8.26. The correct code is

```
function [Delta, CI] = BlsDeltaMC(S0,K,r,T,sigma,dS,NRepl)
nuT = (r - 0.5*sigma^2)*T;
siT = sigma * sqrt(T);
S1 = S0 - dS;
S2 = S0 + dS;
Veps = randn(NRepl,1);
Payoff1 = max(0, S1*exp(nuT+siT*Veps)-K);
Payoff2 = max(0, S2*exp(nuT+siT*Veps)-K);
SampleDiff = exp(-r*T)*(Payoff2 - Payoff1)/2/dS;
[Delta, dummy, CI] = normfit(SampleDiff);
```

Page 478, Eq. (9.3). There are a couple of wrong subscripts, and the equation should read

$$f_{i,j-1} = a_i^* f_{i-1,j} + b_i^* f_{i,j} + c_i^* f_{i+1,j}$$
$$j = N-1, N-2, \ldots, 1, 0;\ i = 1, 2, \ldots, M-1$$

Page 480, last MATLAB line. Replace `EuPutExpl1` with `EuPutExpl`

Page 483, line -6. Replace "time layer i" with "time layer j."

Page 484, MATLAB code in Fig. 9.3. One could wonder why in the LU decomposition here the permutation matrix P is not used. In general, this may result in non-triangular matrices (see MATLAB documentation). Here we trust the tridiagonal nature of the coefficient matrix, which should avoid the need for pivoting.

Page 488, first displayed equation. As usual, the expressions should be intended as a variable assignment in an algorithm. However, we might replace

$$f_{ij} = \max[f_{ij}, K - i\delta S]$$

with

$$f_{ij} = \max[f^c_{ij}, K - i\delta S]$$

to distinguish the option value f_{ij} from the continuation value f^c_{ij}.

Page 490, MATLAB code in Fig. 9.5. The lower boundary value

```
boundval = K*exp(-r*dt*(N-vetj));
```

is wrong. This is an American-style option, so we may immediately grab the intrinsic value K when the underlying asset price is zero, unlike the European-style counterpart. Hence, we should replace the condition with

```
boundval = K;
```

Page 525, Chapter 525. For some reason, I wrote “Linear Stochastic Programming,” but the commonly used term is “Stochastic Linear Programming.”

Page 587, line 11. Replace “Whet” with “What”