Introduction to Fast Fourier Transform in Finance

ALEŠ ČERNÝ

ALEŠ ČERNÝ
is a lecturer in finance at Tanaka Business School, Imperial College London in the U.K.
a.cerny@imperial.ac.uk

The Fourier transform is an important tool in financial economics. It delivers real-time pricing while allowing for a realistic structure of asset returns, taking into account excess kurtosis and stochastic volatility. Fourier transform is also rather abstract and thus intimidating to many practitioners.

This article explains the working of the fast Fourier transform in the familiar binomial option pricing model. In fact, a good understanding of FFT requires no more than some high school mathematics and familiarity with roulette, or a bicycle wheel, or a similar circular object divided into equal-sized segments. The returns to such a small intellectual investment are overwhelming.

The Fourier transform is becoming an increasingly popular and important tool in financial economics because it delivers real-time pricing while allowing for important properties of asset returns, such as excess kurtosis, stochastic volatility, and leverage effects, discussed in Heston [1993], Carr and Madan [1999], and Carr and Wu [2004]. These impressive results come at a price—considerable abstraction that can deter practitioners.

My aim is to explain the working of the discrete Fourier transform (DFT) and its fast implementation (FFT) in the familiar binomial option pricing model. The binomial model plays two roles in this regard. It highlights the usefulness of FFT in an accessible way, showing an important computational tool that has many other applications in finance. It also motivates the passage to continuous time, thereby providing intuition behind fast pricing formulas in a very rich class of models.

I do not provide an exhaustive account of DFT in finance. One can quite easily extend the Fourier pricing formula from a binomial lattice to multinomial lattices; see Černý [2004, Chapter 12]. Further applications of FFT appear in Albanese, Jackson, and Wiberg [2004], Andreas et al. [2002], Benhamou [2002], Chiarella and El-Hassan [1997], Dempster and Hong [2002], and Rebonato and Cooper [1998]. For the most up-to-date developments in option pricing using (continuous) Fourier transform, see Carr and Wu [2004], and for evaluation of hedging errors see Černý [2003] and Hubalek, Kallsen, and Krawczyk [2004].

I. DISCRETE FOURIER TRANSFORM AND BINOMIAL OPTION PRICING

In explaining how and why option prices in the binomial model can be computed via a discrete Fourier transform, we assume the reader is familiar with the concept of risk-neutral pricing. We first introduce complex numbers and discuss their geometric properties, especially as they regard the unit circle; then we define the discrete Fourier transform (DFT) and highlight some of its properties. A simple binomial option pricing example shows how the pricing procedure can be performed on a circle.

Introduction to Complex Numbers

The discrete Fourier transform is about *evenly spaced points on a circle*. From a mathematical point of view, evenly distributed points on a circle are most easily described by complex numbers. The geometry of those numbers determines the properties of Fourier transform.

Complex numbers are a convenient way to capture vectors in a two-dimensional space. Exhibit 1 depicts a vector 2 + i. This is a point on the plane if we move two units on the *real* (horizontal) *axis* and one unit on the *imaginary* (vertical) *axis*.

This terminology is somewhat unfortunate. The imaginary axis is no less real than the real axis. It would be more appropriate to talk about "horizontal" and "vertical" numbers.

The rules for addition of complex numbers are the same as with vectors. For example:

$$\begin{bmatrix} 2 \\ 1 \end{bmatrix} + \begin{bmatrix} 3 \\ -4 \end{bmatrix} = \begin{bmatrix} 5 \\ -3 \end{bmatrix}$$

translated into complex notation would read:

$$(2 + i) + (3 - 4i) = 5 - 3i$$

Likewise, multiplication by a scalar (a real number) works the same way as for vectors:

$$-3 \begin{bmatrix} 2 \\ 1 \end{bmatrix} = \begin{bmatrix} -6 \\ -3 \end{bmatrix}$$

translates into complex numbers as

$$-3(2 + i) = -6 - 3i$$

Complex Multiplication

Complex numbers are very good at describing the movement around a unit circle. As shown in Exhibit 2-A, the unit circle intersects the real axis at points –1, 1, and the imaginary axis at points –i and i. A point A on the unit circle is uniquely characterized by its argument φ—the angle between the real axis and the line OA. More specifically, Exhibit 2-B shows that the point A can be expressed as $\cos \varphi + i \sin \varphi$.

On most computers the functions sin and cos are implemented in such a way that the angle φ must be given in *radians*. Radians measure the distance traveled on the perimeter of the unit circle. The entire perimeter of the unit circle has length 2π, which corresponds to 360°. The angle corresponding to i is 90° or $\pi/2$. The angle corresponding to –1 is 180° or π, and so on, as shown in Exhibit 3.

Below we state four important facts about complex multiplication:

- Multiplying complex numbers on a unit circle requires adding angles. The angle of i is 90°. The angle of i × i will be 90° + 90° = 180°, which corresponds to –1; see Exhibit 4-A. In complex number notation, this gives the formula:

EXHIBIT 1
Complex Number as a Two-Dimensional Vector

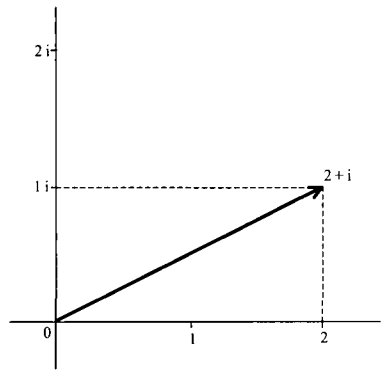


EXHIBIT 2
Point on Unit Circle Expressed as a Complex Number

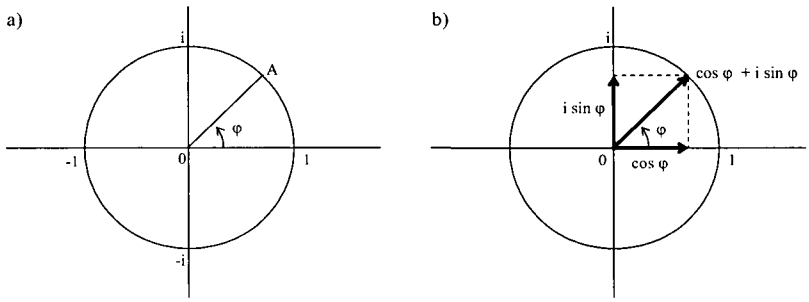


EXHIBIT 3

Conversion Between Degrees and Radians

Angle in degrees	0	30	60	90	180	270	360
Angle in radians	0	$\frac{\pi}{6}$	$\frac{\pi}{3}$	$\frac{\pi}{2}$	π	$\frac{3}{2}\pi$	2π

$$i \times i = i^2 = -1 \quad (1)$$

- The general definition of complex multiplication follows naturally:

$$(a_1 + ib_1) \times (a_2 + ib_2) = a_1a_2 + i(b_1a_2 + a_1b_2) + b_1b_2i^2$$

$$= a_1a_2 - b_1b_2 + i(b_1a_2 + a_1b_2) \quad (2)$$

- It also follows that the "multiplication is adding angles" rule works quite generally on the unit circle:

$$(\cos\varphi_1 + i\sin\varphi_1) \times (\cos\varphi_2 + i\sin\varphi_2)$$
$$= \cos(\varphi_1 + \varphi_2) + i\sin(\varphi_1 + \varphi_2) \quad (3)$$

- One can express points on the unit circle more elegantly using the Euler formula:

$$\cos\varphi + i\sin\varphi = e^{i\varphi} \quad (4)$$

whereby (3) becomes

$$e^{i\varphi_1} \times e^{i\varphi_2} = e^{i(\varphi_1+\varphi_2)} \quad (5)$$

See Exhibit 4-B.

Geometry of Spoked Wheels

It is easy to construct a wheel with evenly placed spokes using complex numbers. Suppose we want to place five evenly spaced points on the unit circle. One-fifth of the full circle is characterized by the angle $2\pi/5$. Hence, the first spoke will be placed at $e^{i\frac{2\pi}{5}}$. Let us denote this number by z_5 (fifth root of unity):

$$z_5 \equiv e^{i\frac{2\pi}{5}}$$

Since the multiplication by z_5 causes counterclockwise rotation by one-fifth of the full circle, the second spoke will be $(z_5)^2$, the third spoke $(z_5)^3$, and so on, as Exhibit 5-A shows.

This provides a natural numbering of the spokes, according to the number of elementary rotations needed to reach the particular spoke. Note that since we are moving in a circle we will come back to the starting point after five rotations counterclockwise:

$$(z_5)^0 = (z_5)^5 = (z_5)^{10} = (z_5)^{15} = \ldots$$
$$(z_5)^1 = (z_5)^6 = (z_5)^{11} = (z_5)^{16} = \ldots$$

and also after five rotations clockwise:

$$(z_5)^0 = (z_5)^{-5} = (z_5)^{-10} = (z_5)^{-15} = \ldots$$

$$(z_5)^1 = (z_5)^{-4} = (z_5)^{-9} = (z_5)^{-14} = \ldots$$

Thus the numbering of spokes is ambiguous. For example, indexes 0, 5, –5 refer to the same spoke, as in Exhibit 5-B.

Below we summarize the most important properties of evenly spaced points on the unit circle. These properties are essential for the understanding of the discrete Fourier transform.

- Let z_n be a rotation by one n-th of a full circle

$$z_n \equiv e^{i\frac{2\pi}{n}}$$

Then:

$$(z_n)^0 + (z_n)^1 + \ldots + (z_n)^{n-1} = 0 \quad (6)$$

for any n. This is because the points $(z_n)^0$, $(z_n)^1$, . . . , $(z_n)^{n-1}$ are evenly distributed on a unit circle, and thus the result of summation must not change if we rotate the set of points by one n-th of a full circle. The only vector that remains unchanged after such rotation is the zero vector.

- One can generalize this result further. Let k be an integer between 1 and $n - 1$. Then:

$$\left(z_n^k\right)^0 + \left(z_n^k\right)^1 + \ldots + \left(z_n^k\right)^{n-1} = 0 \quad (7)$$

for any n.

The reason for this result is again rotational symmetry of points $\left(z_n^k\right)^0, \left(z_n^k\right)^1, \ldots, \left(z_n^k\right)^{n-1}$. The difference is that in the Equation (6) sequence each spoke occurs *exactly once*, while in the Equation (7) sequence the same spoke can occur several times (try $n = 4$, $k = 2$).

EXHIBIT 4
Complex Multiplication on a Unit Circle

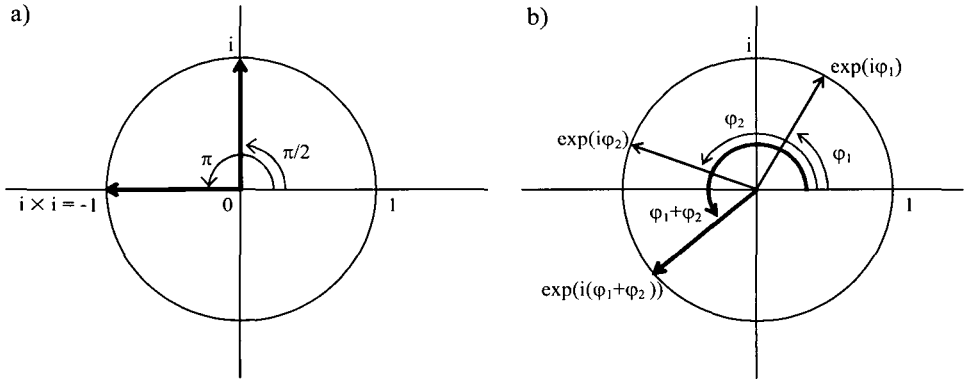
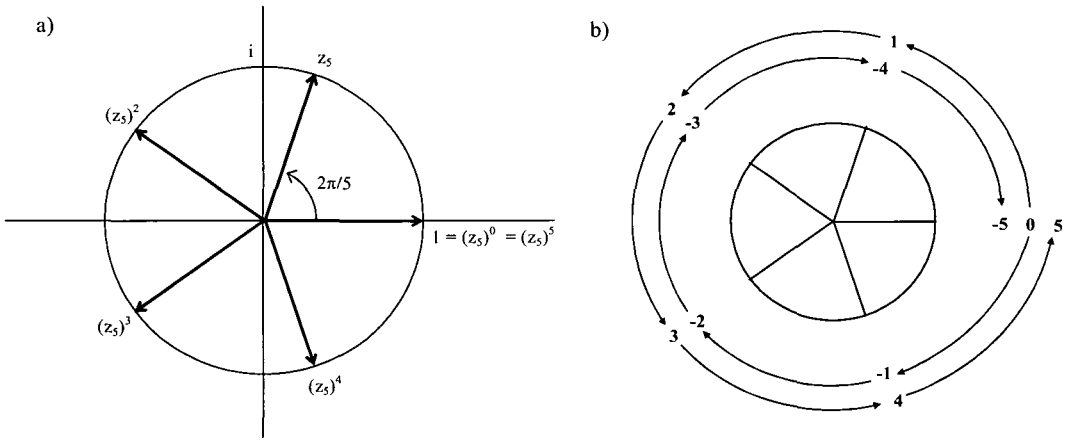


EXHIBIT 5
Evenly Distributed Points on a Circle



- The case with $k = 0$ requires special attention. Since $(z_n^0)^j = 1$ for all j we have:

$$\left(z_n^k\right)^0 + \left(z_n^k\right)^1 + \ldots + \left(z_n^k\right)^{n-1} = n$$

To summarize:

$$\left(z_n^k\right)^0 + \left(z_n^k\right)^1 + \ldots + \left(z_n^k\right)^{n-1} = n \text{ for } k = 0, \pm n, \pm 2n, \ldots (8)$$

$$\left(z_n^k\right)^0 + \left(z_n^k\right)^1 + \ldots + \left(z_n^k\right)^{n-1} = 0 \text{ for } k = 0, \pm n, \pm 2n, \ldots (9)$$

Reverse Order on a Circle

Given a sequence of n numbers $a = [a_0, a_1 \ldots, a_{n-1}]$, we can say that

$$\text{rev}(a) \equiv [a_0, a_{n-1}, \ldots, a_1]$$

is a in *reverse order.* If a is written around a circle in *counterclockwise* direction then rev(a) is found by reading from a_0 in clockwise direction. See Exhibit 6. Note that rev(a) *is not* equal to $[a_{n-1}, \ldots, a_1, a_0]$.

We will require the following fact for future reference. For any k, the sequence

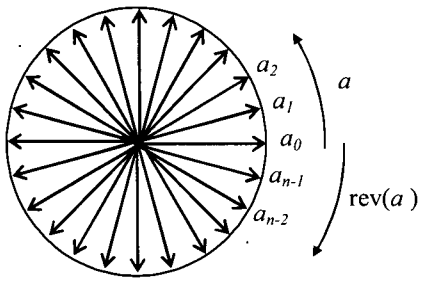
$$\left(z_n^k\right)^0, \left(z_n^k\right)^1, \ldots, \left(z_n^k\right)^{n-1} = n.$$

is the same as the sequence

$$\left(z_n^{-k}\right)^0, \left(z_n^{-k}\right)^1, \ldots, \left(z_n^{-k}\right)^{n-1}$$

taken in the reverse order:

EXHIBIT 6
Reverse Order on a Circle



$$\text{rev}\left(\left(z_n^{-k}\right)^0, \left(z_n^{-k}\right)^1, \ldots, \left(z_n^{-k}\right)^{n-1}\right) = \left(z_n^k\right)^0, \left(z_n^k\right)^1, \ldots, \left(z_n^k\right)^{n-1} \tag{10}$$

This is because $\left(z_n^{-k}\right)^{n-j} = z_n^{-kn+kj} = z_n^{kj} = \left(z_n^k\right)^j$ for any j.

Discrete Fourier Transform (DFT)

Now take $z_n \equiv e^{i\frac{2\pi}{n}}$ (this number is called the n-th root of unity). Let $a_0, a_1, \ldots, a_{n-1}$ be a sequence of n (in general complex) numbers. The *discrete Fourier transform* of $a_0, a_1, \ldots, a_{n-1}$ is the sequence $b_0, b_1, \ldots, b_{n-1}$ such that:

$$b_k = \frac{a_0\left(z_n^k\right)^0 + a_1\left(z_n^k\right)^1 + \ldots + a_{n-1}\left(z_n^k\right)^{n-1}}{\sqrt{n}} = \frac{1}{\sqrt{n}}\sum_{j=0}^{n-1} a_j z_n^{jk} = \frac{1}{\sqrt{n}}\sum_{j=0}^{n-1} a_j e^{i\frac{2\pi}{n}jk} \tag{11}$$

In short we will write:

$$\mathcal{F}(a) = b$$

Equation (11) represents the *forward transform*. The *inverse transform* is

$$\tilde{a}_l = \frac{\tilde{b}_0\left(z_n^{-l}\right)^0 + \tilde{b}_1\left(z_n^{-l}\right)^1 + \ldots + \tilde{b}_{n-1}\left(z_n^{-l}\right)^{n-1}}{\sqrt{n}} = \frac{1}{\sqrt{n}}\sum_{k=0}^{n-1} \tilde{b}_k z_n^{-kl} = \frac{1}{\sqrt{n}}\sum_{k=0}^{n-1} \tilde{b}_k e^{-i\frac{2\pi}{n}kl} \tag{12}$$

which we can write:

$$\tilde{a} = \mathcal{F}^{-1}\left(\tilde{b}\right)$$

- The inverse discrete Fourier transform of sequence $\tilde{b}_0, \tilde{b}_1, \ldots, \tilde{b}_{n-1}$ is the same as the forward transform of the same sequence in reverse order:
$$\mathcal{F}^{-1}(\tilde{b}) = \mathcal{F}(\text{rev}(\tilde{b})) \tag{13}$$

and vice versa:

$$\mathcal{F}^{-1}(\text{rev}(\tilde{b})) = \mathcal{F}(\tilde{b}) \tag{14}$$

This is a direct consequence of (10).

- $\mathcal{F}^{-1}$ is indeed an inverse transformation of $\mathcal{F}$; that is:

$$\mathcal{F}^{-1}(\mathcal{F}(a)) = \mathcal{F}(\mathcal{F}^{-1}(a)) = a \tag{15}$$

This result relies on (8) and (9); for a proof see the appendix.

Binomial Option Pricing

Consider the distribution of FTSE 100 monthly returns calibrated to reflect market volatility of 4.4% a month and expected rate of return of 0.9% a month:

$$pR_u + (1-p)R_d = 1.009$$
$$pR_u^2 + (1-p)R_d^2 = 0.044^2 + 1.009^2$$

Choosing the objective probability to be $p = ½$, we solve for R_u and R_d:

$$R_u = 1.053 \text{ with } p_u = \frac{1}{2} \tag{16}$$

$$R_d = 0.965 \text{ with } p_d = \frac{1}{2} \tag{17}$$

Assuming that the initial value of the FTSE index is 5100 points, the evolution of the index in the three months ahead is given by the lattice in Exhibit 7. Suppose we want to price a call option struck at $K = 5355$ (5% out of the money), maturing three months from now. The intrinsic value of the option at maturity is

$$C(3) = \begin{bmatrix} 599.64 \\ 102.00 \\ 0.00 \\ 0.00 \end{bmatrix} \tag{18}$$

EXHIBIT 7
Binomial Stock Price Lattice

number of low returns	$S(0)$	$S(1)$	$S(2)$	$S(3)$
0	5100.00	5370.30	5654.93	5954.64
1		4921.50	5182.34	5457.00
2			4749.25	5000.96
3				4583.02

Asset pricing theory tells us that the no-arbitrage price of the payoff:

$$\begin{bmatrix} C_u \\ C_d \end{bmatrix}$$

is given as the risk-neutral expectation of the discounted payoff:

$$\text{No-Arbitrage Value } (C) = \frac{q_u C_u + q_d C_d}{R_f} \tag{19}$$

where the risk-neutral probabilities q_u and q_d are chosen so that the risk-neutral expected return of all basis assets is equal to the risk-free return:

$$q_u + q_d = 1$$
$$q_u R_u + q_d R_d = R_f$$

The values q_u/R_f and q_d/R_f are known as *state prices*.

Assuming a risk-free rate of 4% per year, the monthly risk-free return is:

$$R_f = 1.04^{1/12} = 1.0033$$

This gives conditional risk-neutral probabilities of

$$q_u = \frac{R_f - R_d}{R_u - R_d} = \frac{1.0033 - 0.965}{1.053 - 0.965} = 0.43523 \tag{20}$$
$$q_d = \frac{R_u - R_f}{R_u - R_d} = \frac{1.053 - 1.0033}{1.053 - 0.965} = 0.56477 \tag{21}$$

and the valuation formula:

$$\text{No-Arbitrage Value } (C) = \frac{0.43523 C_u + 0.56477 C_d}{1.0033} \tag{22}$$

Recursive application of (22) with terminal value (18) produces the option prices in Exhibit 8.

Option Pricing on a Circle

For any two n-dimensional vectors $a = [a_0, a_1, \ldots, a_{n-1}]$, $b = [b_0, b_1, \ldots, b_{n-1}]$ we define *circular (cyclic) convolution* of a and b to be a new vector c:

$$c = a \circledast b$$

such that

$$c_j = \sum_{k=0}^{n-1} a_{j-k} b_k \tag{23}$$

One will immediately note that the index $j-k$ can be negative. If this occurs, we simply add n to get a result between 0 and $n-1$; this practice is consistent with the spoke numbering introduced in Section I, and it merely reflects movement in a circle.

One can graph the circular convolution as follows:

1. Set up two concentric circles divided into n equal segments. Write a around the inner circle clockwise and b around the outer circle counterclockwise. Exhibit 9 shows this for $n = 4$.
2. Perform a scalar multiplication between the two circles. In Exhibit 9, this would give

$$a_0 b_0 + a_3 b_1 + a_2 b_2 + a_1 b_3$$

3. *Turn the inner circle counterclockwise* by $1/n$-th of a full circle. Repeat the scalar multiplication between the circles. The result is c_1. In Exhibit 10:

$$c_1 = a_1 b_0 + a_0 b_1 + a_3 b_2 + a_2 b_3$$

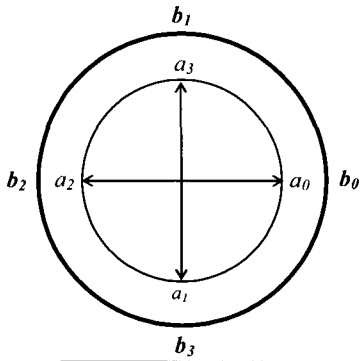
4. Repeat this procedure to compute $c_2, \ldots, c_{n-1}$, each time giving the inner circle $1/n$-th turn counterclockwise.

How can one use the circular convolution for option pricing? If we write both the option payoff and the pricing

EXHIBIT 8
Option Prices in a Binomial Lattice

number of low returns	$C(0)$	$C(1)$	$C(2)$	$C(3)$
0	81.36	162.66	317.54	599.64
1		19.19	44.25	102.00
2			0.00	0.00
3				0.00

EXHIBIT 9
Computing First Element of Circular Convolution $a \circledast b$



kernel in the clockwise direction and then rotate the option payoff in the counterclockwise direction, we will obtain option prices in the natural order from highest to lowest.

To be specific, let us return to the binomial option pricing model. At maturity, the option can have four different values:

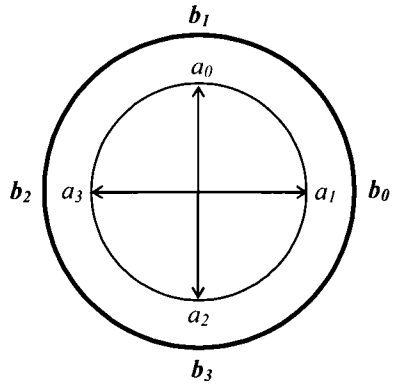
$$C(3) = \begin{bmatrix} 599.64 & 102.00 & 0.00 & 0.00 \end{bmatrix}$$

Let vector q contain the conditional one-period risk-neutral probabilities $q_u = 0.43523$, $q_d = 0.56477$. Since there are just two states over one period, the remaining entries will be padded by zeros:

$$q = \begin{bmatrix} q_u & q_d & 0.00 & 0.00 \end{bmatrix}$$

Finally, recall that the risk-free return is $R_f = 1.0033$. Thus, to compute option prices at time $t = 2$ we need to evaluate

EXHIBIT 10
Computing Second Element of Circular Convolution $a \circledast b$



$$c = C(3) \circledast \text{rev}(q/R_f)$$

This operation is depicted graphically in Exhibit 11, where the option payoffs $C(3)$ are on the inner circle, and the state prices q/R_f are on the outer circle, both written in clockwise direction. Numerically we obtain:

$$C(3) \circledast \text{rev}(q/R_f) = \begin{bmatrix} 317.54 & 44.25 & 0 & 337.54 \end{bmatrix} \quad (24)$$

Note that we need only the first three prices in (24). The last entry is meaningless—it corresponds to the no-arbitrage price of the payoff [0 599.64].

This result can be generalized as follows. Consider a binomial model where $C(j)$ is the vector of option prices at date $j = 0, 1, \ldots, N$. Denote by q the vector containing the risk-neutral probabilities q_u and q_d, padded by zeros to have the same dimension as $C(N)$. By backward substitution:

$$\begin{aligned} C(N-1) &= C(N) \circledast \text{rev}(q)/R_f, \\ C(N-2) &= C(N) \circledast \text{rev}(q) \circledast \text{rev}(q)/R_f^2, \\ C(j) &= C(N) \circledast \overbrace{\text{rev}(q) \circledast \text{rev}(q) \circledast \ldots \circledast \text{rev}(q)}^{(N-j) \text{ times}}/R_f^j \end{aligned} \quad (25)$$

The vectors $C(j)$ computed in this manner have more entries than needed. The useful $j + 1$ entries are at the top end of each vector.

Numerical results are reported in Exhibit 12. Compare the relevant boldfaced entries with those in Exhibit 8.

Circular Pricing via Discrete Fourier Transform

Now we reformulate the circular pricing formula in Equation (25) using the discrete Fourier transform. The discrete Fourier transform has one very useful property—it turns circular convolutions into products:

$$\mathcal{F}(a \circledast b) = \sqrt{n}\mathcal{F}(a)\mathcal{F}(b) \quad (26)$$

$$\mathcal{F}^{-1}(a \circledast b) = \sqrt{n}\mathcal{F}^{-1}(a)\mathcal{F}^{-1}(b) \quad (27)$$

$$n = \text{dimension of } a \quad (28)$$

The appendix provides a proof. This property can be used to great advantage in pricing.

Recall from Equation (25) that

$$C_0 = C_N \circledast \overbrace{\text{rev}(q) \circledast \text{rev}(q) \circledast \ldots \circledast \text{rev}(q)}^{N \text{ times}} / R_f^N$$

where N is the number of time periods to maturity. Now apply the inverse transform $\mathcal{F}^{-1}$ to both sides, using property (27) on the right-hand side:

$$\mathcal{F}^{-1}(C_0) = \mathcal{F}^{-1}(C_N) \times \left(\sqrt{\text{dimension of } C_N}\mathcal{F}^{-1}(\text{rev}(q))/R_f\right)^N \quad (29)$$

In a binomial model, the dimension of C_N is $N + 1$. Furthermore, recall from (14) that $\mathcal{F}^{-1}(\text{rev}(q)) = \mathcal{F}(q)$ and substitute this into (29):

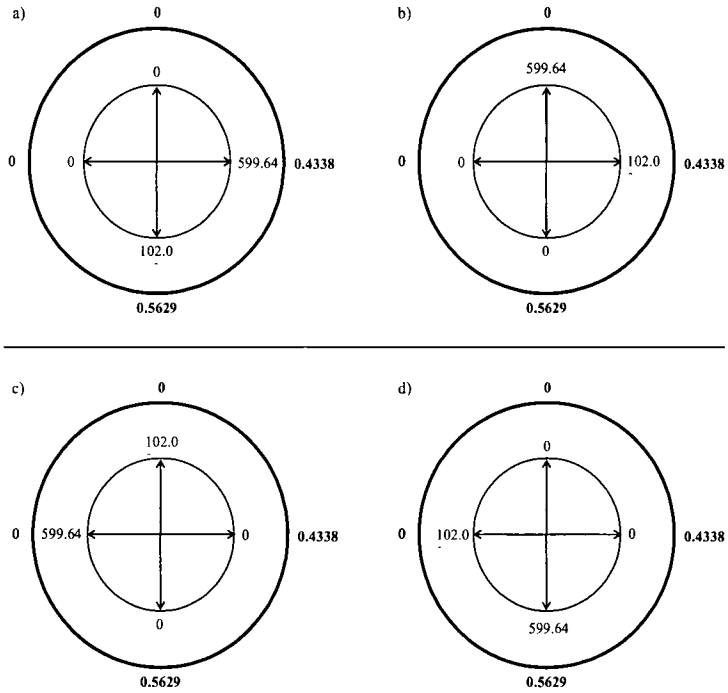
$$\mathcal{F}^{-1}(C_0) = \mathcal{F}^{-1}(C_N) \times \left(\sqrt{N+1}\mathcal{F}(q)/R_f\right)^N$$

Finally, apply the forward transform to both sides again, and use (15) on the left-hand side:

$$C_0 = \mathcal{F}\left(\mathcal{F}^{-1}(C_N) \times \left(\sqrt{N+1}\mathcal{F}(q)/R_f\right)^N\right)$$

We have provided the following theorem. Consider a model with independent and identically distributed (IID) stock returns and constant interest rate, represented by a recombining binomial tree with N periods and $N + 1$ trading dates. Let the $(N+1)$-dimensional vector C_N be the payoff of the option at expiration. Let q contain the one-step risk-neutral probabilities as the first two entries, with the remaining $N - 1$ entries being zeros. Then the first element of $(N + 1)$-dimensional vector C_0:

EXHIBIT 11
Option Pricing on a Circle



$$C_0 = \mathcal{F}\left(\mathcal{F}^{-1}(C_N) \times \left(\sqrt{N+1}\mathcal{F}(q)/R_f\right)^N\right) \quad (30)$$

is the no-arbitrage price of the option at time 0.

The role of the forward and inverse transforms is symmetrical. That is, we also have:

$$C_0 = \mathcal{F}^{-1}\left(\mathcal{F}(C_N) \times \left(\sqrt{N+1}\mathcal{F}^{-1}(q)/R_f\right)^N\right) \quad (31)$$

II. FAST FOURIER TRANSFORM (FFT)

We can implement the pricing formula in Equation (30) on a computer using fast DFT routines, known as FFTs. It is highly unlikely any reader would want to write his or her own DFT code, for this would be counterproductive, given the wealth and the level of specialization of ready-made algorithms. Prepackaged algorithms save time, but with little documentation at hand, implementing otherwise sound mathematical formulas may not prove straightforward. We can provide guidelines that ensure a trouble-free transition between the theoretical pricing formula (30) and a computer code using a DFT routine of the reader's choice. Specific examples are given in GAUSS and Matlab.

There are two main issues in the use of (fast) DFT

routines: 1) finding out the mathematical definition of a specific DFT routine, and 2) choosing the right input length for fast computation.

Mathematical Definitions

Every textbook, and indeed every computer language, defines the forward and inverse transforms slightly differently. Thus the first task of any user is to find out how a given computer routine, call it dft, is related to the theoretical transforms $\mathcal{F}$ and $\mathcal{F}^{-1}$ defined in (11) and (12). To do so, one proceeds in two simple steps.

In the first step, one determines the normalization factor. Define a = [1 0 0 0] and compute $\tilde{a}$ = dft(a). If $\tilde{a}_0$ = 0.25, then:

either dft = $\mathcal{F}/\sqrt{n}$

or dft = $\mathcal{F}^{-1}/\sqrt{n}$

If $\tilde{a}_0$ = 0.5, then:

either dft = $\mathcal{F}$

or dft = $\mathcal{F}^{-1}$

If $\tilde{a}_0$ = 1, then:

either dft = $\sqrt{n}\mathcal{F}$

or dft = $\sqrt{n}\mathcal{F}^{-1}$

Second, to ascertain whether one is dealing with a forward or an inverse transform, one defines b = [0 1 0 0] and evaluates $\tilde{b}$ = dft (b). If the imaginary part of $\tilde{b}_1$ is positive, then dft is proportional to $\mathcal{F}$; otherwise it is proportional to $\mathcal{F}^{-1}$.

In the case of the lattice pricing formulas (30) and (31), one will use two routines, say, dft and dfti, which are inverse to each other. In this case, it does not really matter which of the two transforms is forward and which is inverse. But in other applications in Section III below, it is absolutely crucial to know whether a given routine is proportional to $\mathcal{F}$ or $\mathcal{F}^{-1}$.

Example 1 provides an illustration. In GAUSS, the two DFT transforms are called dfft and dffti, respectively, and they are related to $\mathcal{F}$ and $\mathcal{F}^{-1}$ as follows:

EXHIBIT 12

Option Prices Obtained from Circular Pricing Equation (25)

number of low returns	$C(0)$	$C(1)$	$C(2)$	$C(3)$
0	**81.36**	**162.66**	**317.54**	**599.64**
1	115.28	**19.19**	**44.25**	**102.00**
2	265.47	190.01	**0.00**	**0.00**
3	232.62	325.17	337.54	**0.00**

Boldfaced entries are the relevant ones.

$$\texttt{dfft}(a) \equiv \frac{\mathcal{F}^{-1}(a)}{\sqrt{n}}$$

$$\texttt{dffti}(a) \equiv \sqrt{n}\mathcal{F}(a)$$

where n is the dimension of vector a.

Equation (30) therefore becomes:

$$C_0 = \texttt{dffti}\left(\texttt{dfft}(C_N) \times (\texttt{dffti}(b))^N\right) \quad (32)$$

Suppose the vectors C_ N and b have already been defined in GAUSS. To compute the option price at t = 0 we would use the code:

```
C_0 = dffti( dfft(C_N).*(dfft(b)^N) );
print ''no-arbitrage price at t=0 is '' C_0[1];
```
(33)

The .* command stands for element-by-element multiplication.

The DFT algorithm is approximately three times faster than backward recursion in the binomial model. The computation time for both algorithms grows quadratically with the number of periods, as shown in Exhibit 13.[1]

Input Length

A naive implementation of the DFT algorithm with n-dimensional input requires n^2 complex multiplications. An efficient implementation of DFT, known as the fast Fourier transform (FFT), will require only Kn ln n operations, but one still has to choose n carefully because the constant K can be very large for some choices of n.[2]

Some FFT implementations automatically restrict the transform length to the most suitable values of n (typically $n = 2^p$ or $n = 2^p 3^q 5^r$), which is the case in GAUSS. Others, such as Matlab, will compute FFT of any length; here it is

particularly important for the user to choose n sensibly, or the FFT algorithm may turn out to be very slow indeed.

Example 2. The forward and the inverse FFT in Matlab are called fft and ifft, respectively:

$$\texttt{fft}(a) \equiv \sqrt{n}\mathcal{F}^{-1}(a) \tag{34}$$

$$\texttt{ifft}(a) \equiv \frac{\mathcal{F}(a)}{\sqrt{n}} \tag{35}$$

where n is the dimension of vector a. The option pricing Equation (30) therefore becomes:

$$C_0 = \texttt{ifft}\left(\texttt{fft}(C_N) \times ((N+1) \times \texttt{ifft}(b))^N\right)$$

which in terms of Matlab code reads:

```
C_0 = ifft( fft(C_N).*(((N+1)*ifft(b)).^N) );
sprintf 'no-arbitrage price at t=0 is %0.2f' C_0(1);
```
(36)

The commands .* and .^ stand for element-by-element multiplication and exponentiation, respectively.

In many instances, FFT of length n_1 is faster than FFT of length n_2 even though $n_1 > n_2$. This somewhat counterintuitive phenomenon is illustrated in Exhibit 14.

To understand why some transform lengths are more suitable than others, we need one piece of terminology and one fact: 1) the FFT algorithm for length $n = 2^p$ is called a radix-2 algorithm; 2) the higher the b, the slower the radix-b algorithm per output length. There is one notable exception: Radix-4 is faster than radix-2 by about 25%.

In practice, one uses transforms of size $n = 2^p3^q5^r$. If the original vector size is *not* of this form, the appropriate number of zeros is added. Ideally, q and r should be small compared to p.

The advantage of using mixed-radix algorithms is twofold: 1) more transform lengths are available, which means one need not pad the input with too many zeros, and 2) one can use the operation-saving prime factor algorithm.[3]

With vector size $2^{10} + 1 = 1025$, the next available size for the radix-2 algorithm is $n = 2048 = 2^{11}$ but with a mixed 2, 3, 5-radix algorithm one could use length $n = 1080 = 2^3 3^3 5$, which is nearly half the size, so the Fourier transform evaluation is twice as fast as the radix-2 algorithm.

Note that highly composite lengths such as 1080000 $= 2^6 3^3 5^4$ are calculated more quickly than simple powers of similar length such as $2^{20} = 1048576$ (*see Exhibit 14*). Transforms that are not of the length $n = 2^p3^q5^r$ can take a long time to compute, especially if n is a large prime; again see Exhibit 14.

Example 3. Matlab will allow the user to perform FFT of any length; this is done using commands (34) and (35). As we have noted above, however, it is sensible to restrict transform lengths to $n = 2^p3^q5^r$ with q and r small relative to p to obtain the best performance. Matlab provides function nextpow2 giving the next higher power of 2. It also allows the user to specify the transform length by

EXHIBIT 13
Comparison of Pricing Speeds

trading interval	number of	execution time in seconds	
in minutes	periods	**DFT**	**backward recursion**
60	504	0.15	0.4
30	1008	0.6	1.6
15	2016	2.3	6.4
5	6048	20.8	61.6

Pentium III 750MHz, 128Mb RAM, GAUSS

EXHIBIT 14
Execution Time for Different Input Lengths

n	factorization	execution time in seconds
499 979	499 979	27.2
1048 575	$3 \times 5^2 \times 11 \times 31 \times 41$	5.2
1048 576	2^{20}	0.93
1080 000	$2^6 3^3 5^4$	0.11

Pentium III 750MHz, 128Mb RAM, Matlab.

EXHIBIT 15
Speed of Binomial Pricing Using DFT and FFT Algorithms

trading interval	number of	execution time in seconds	
in minutes	periods	**DFT**	**FFT**
30	1008	0.6	0.003
15	2016	2.3	0.006
5	6048	20.8	0.022
1	30240	510	0.27

Pentium III 750MHz, 128Mb RAM, GAUSS

EXHIBIT 16

Option Price and Option Delta—Continuous-Time and Binomial Approximation

				Black–Scholes
△t (seconds)	60	10	1	0
Option price	75.93398	75.93284	75.93286	75.93288
Option delta	0.31668534	0.31668346	0.31668334	0.31668331

including it as a second optional argument of fft and ifft.

Hence a fast implementation of (36) in Matlab would read:

```
length = 2^nextpow2(N+1);
C_0 = ifft( fft(C_N,length)
.*((length*ifft(b,length)).^N) );
```

The padding of the original input C_N by zeros to the dimension length is done automatically.

To find the nearest transform length of the form $n = 2^p3^q5^r$, one can use the code:

```
length = N+1;
while max(factor(length)) > 5;
    length = length+1;
end;
```

Example 4. In GAUSS the fast Fourier forward and inverse transforms are performed by functions fftn and ffti. These functions use Temperton's [1992] mixed 2, 3, 5-radix algorithm, and the padding of input vector by zeros to the nearest available length $n = 2^p3^q5^r$ is done automatically. If n is the input dimension, the output dimension from fftn and ffti will be nextn(n).

In terms of GAUSS code, one writes as in (33):

```
C_0 = ffti( fftn(C_N).*(ffti(b)^N) );
```

One can increase the speed further by choosing a composite length $n = 2^p3^q5^r$ where q and r are non-zero but small relative to p. The optimal length is given by GAUSS function optn($N + 1$), and the padding by zeros to this dimension must be performed by the user.

The FFT implementation of the binomial pricing algorithm is blazingly fast compared to the DFT, as Exhibit 15 shows.[4]

Because it is so fast, one can explore higher trading frequencies and see that the Black-Scholes formula really does describe the limiting value (*see Exhibit 16*). Note that the Black-Scholes formula itself is still about 10,000 times faster than the FFT algorithm.

III. FURTHER APPLICATIONS OF FFT IN FINANCE

Practical applications of the discrete or the fast Fourier transform in modern finance go beyond the binomial model, but the essential structure of the pricing formulas is that of Equation (30). To motivate the passage to continuous time, we can rewrite the DFT pricing Equation (30) to take explicit account of the maturity date T and the rebalancing frequency Δt, with $N_{\Delta t} = T/\Delta t$ trading periods and instantaneous risk-free rate r:

$$\begin{aligned} C_0 &= \mathcal{F}\left(\mathcal{F}^{-1}\left(C_{T,\Delta t}\right) \times \left(\sqrt{N_{\Delta t}+1}\mathcal{F}\left(q_{\Delta t}\right)\mathrm{e}^{-r\Delta t}\right)^{N_{\Delta t}}\right) \\ &= \mathrm{e}^{-rT}\mathcal{F}\left(\mathcal{F}^{-1}\left(C_{T,\Delta t}\right) \times \left(\sqrt{N_{\Delta t}+1}\mathcal{F}\left(q_{\Delta t}\right)\right)^{N_{\Delta t}}\right) \end{aligned} \tag{37}$$

The quantity

$$\left(\sqrt{N_{\Delta t}+1}\mathcal{F}\left(q_{\Delta t}\right)\right)^{N_{\Delta t}}$$

is known as the (risk-neutral) *characteristic function* of the log stock price. In practice we are mainly interested in models where the continuous-time limit of (37) is available in closed form. This is the case in the class of exponential Lévy models with an affine stochastic volatility process, as discussed in Carr and Wu [2004]. This class includes a large number of popular models allowing for excess kurtosis, stochastic volatility, and leverage effects. It includes, among others, the stochastic volatility models of Heston [1993], Duffie, Pan, and Singleton [2000], and all exponential Lévy models (see, for example, Madan

and Seneta [1990], and Eberlein, Keller, and Prause [1998]). For an exhaustive characterization of affine processes, see Duffie, Filipović, and Schachermayer [2003].

In the continuous-time limit, the discrete Fourier transform is replaced by the (continuous) Fourier transform. That is, we wish to find coefficients $\psi(v)$ such that:

$$C_T(\ln S_T) = \int_{\beta-i\infty}^{\beta+i\infty} \psi(v) e^{iv \ln S_T} dv \tag{38}$$

for some real constant β.[5]

The recipe for obtaining the coefficients $\psi(v)$ is known—it is given by the inverse Fourier transform:[6]

$$\psi(v) = \frac{1}{2\pi} \int_{-\beta-i\infty}^{-\beta+i\infty} C_T(\ln S_T) e^{-iv \ln S_T} d \ln S_t \tag{39}$$

For example, a simple calculation in Carr and Madan [1999] shows that the coefficients ψ of a call option with strike price e^k take the form:

$$\psi(v) = \frac{e^{-(v-1)k}}{2\pi v(v-1)} \text{ for } \operatorname{Re} v > 1$$

Substituting for C_T from (38), the risk-neutral pricing formula reads:

$$\begin{aligned} C_0(\ln S_0) &= e^{-rT} E^Q [C_T(\ln S_T)] \\ &= e^{-rT} E^Q \left[\int_{\beta-i\infty}^{\beta+i\infty} \psi(v) e^{iv \ln S_T} dv \right] \\ &= e^{-rT} \int_{\beta-i\infty}^{\beta+i\infty} \psi(v) E^Q \left[e^{iv \ln S_T} \right] dv \end{aligned} \tag{40}$$

where $E^Q[e^{iv \ln S_T}]$ is the risk-neutral characteristic function of the log stock price.

It should now be clear that the continuous-time pricing formula (40) is a direct analogue of its discrete-time counterpart (37), where instead of the discrete characteristic function $(\sqrt{N_{\Delta t}+1}\mathcal{F}(q_{\Delta t}))^{N_{\Delta t}}$ we use the continuous characteristic function $E^Q[e^{iv \ln S_T}]$. Instead of discrete Fourier coefficients $\mathcal{F}^{-1}(C_{T,\Delta t})$ we use the continuous coefficients ψ, and instead of summation we use integration.

There is nevertheless one major difference between (37) and (40). While the former spends a significant amount of time computing the characteristic function of log returns and Fourier coefficients of the option, the latter provides both quantities in closed form. This makes the continuous-time pricing formula (40) even faster than the accelerated binomial formula (30).

Example 5

In the Heston [1993] model:

$$\begin{aligned} d\sigma_t^2 &= (a - b\sigma_t^2)dt + \tilde{\sigma}\sigma_t dB_1^Q \\ d \ln S_t &= \left(r - \frac{\sigma_t^2}{2} \right) dt + \sigma_t dB_2^Q \end{aligned}$$

we have

$$\begin{aligned} \phi^Q(-iv) &= E^Q \left[e^{v \ln(S_T/S_0)} \right] = e^{\gamma(v) + \delta(v)\sigma_0^2}, \\ \gamma(v) &= rTv + \frac{a}{\tilde{\sigma}^2} \left(b - \rho\tilde{\sigma}v - 2 \ln \left(\frac{1 - c_2(v)e^{c_1(v)T}}{1 - c_2(v)} \right) \right) \\ \delta(v) &= \frac{(b - \rho\tilde{\sigma}v + c_1(v)) \left(1 - e^{c_1(v)T} \right)}{\tilde{\sigma}^2 \left(1 - c_2(v)e^{c_1(v)T} \right)} \\ c_1(v) &= \sqrt{(b - \rho\tilde{\sigma}v)^2 - \tilde{\sigma}^2 (v + v^2)} \\ c_2(v) &= \frac{b - \rho\tilde{\sigma}v + c_1(v)}{b - \rho\tilde{\sigma}v - c_1(v)} \end{aligned}$$

where $\rho = \operatorname{Corr}(dB_1^Q, dB_2^Q)$.

Option pricing therefore boils down to evaluation of integrals of the type:

$$\begin{aligned} C_0(\ln S_0) &= S_0 e^{-rT} \int_{\beta-i\infty}^{\beta+i\infty} \frac{e^{(v-1)(\ln S_0 - k)}}{2\pi v(v-1)} \phi^Q(-iv) dv \\ &= 2S_0 e^{-rT} \int_{\beta+i0}^{\beta+i\infty} \operatorname{Re} \left(\frac{e^{(v-1)(\ln S_0 - k)}}{2\pi v(v-1)} \phi^Q(-iv) \right) dv \end{aligned} \tag{41}$$

where both $\psi(v)$ and $\phi^Q(v)$ are known.

To evaluate (41), one truncates the integral at a high value of $\operatorname{Im} v$, and then uses a numerical quadrature to approximate it by a sum; see Lee [2004] for a detailed exposition. This yields an expression of the type:

$$C_0(\ln S_0) \approx 2 \operatorname{Re} \left(S_0 e^{-rT} \sum_{j=0}^{n-1} w_j \frac{e^{(v_j - 1)(\ln S_0 - k)}}{2\pi v_j (v_j - 1)} \phi^Q(-iv_j) \right) \tag{42}$$

where the integration weights w_j and abscissas v_j depend on the quadrature rule. It is particularly convenient to use

Newton-Cotes rules, which employ equidistantly spaced abscissas. For example, a trapezoidal rule yields:

$$\begin{aligned} v_j &= \beta + \mathrm{i}j\triangle v \\ \operatorname{Im} v_{\max} &= (n-1)\triangle v \\ w_0 &= w_n = \frac{1}{2}\triangle v \\ w_1 &= w_2 = \ldots = w_{n-1} = \triangle v \end{aligned} \tag{43}$$

If the characteristic function of log returns is known, one needs to evaluate a single sum (42) to find the option price. Consequently, there is no need to use FFT if one wishes to evaluate the option price for one fixed log strike k.

FFT Option Pricing with Multiple Strikes

The situation is very different if we want to evaluate the option price (42) for many different strikes simultaneously. Let us consider $m = 121$ values of moneyness $\kappa_l = \ln S_0 - k_l$ ranging from -30% to 30% with increment $\Delta\kappa = 0.5\%$:

$$\kappa_l = \kappa_{\max} - l\triangle\kappa, \tag{44}$$

$$\kappa_{\max} = 0.30,\ l = 0, \ldots, m-1 \tag{45}$$

The idea of using FFT in this context is due to Carr and Madan [1999], but it has recently been improved by using the so-called z-transform; see Chourdakis [2004].

The number

$$a_0 z^0 + a_1 z^{-1} + \ldots + a_{n-1} z^{-(n-1)}$$

is called the z-transform of sequence a. The discrete Fourier transform of sequence a is obtained as a special case of the z-transform with n specific values of z:

$$z_l = \mathrm{e}^{-\mathrm{i}\frac{2\pi}{n}l},\ l = 0, 1, \ldots, n-1$$

Carr and Madan have noted that with equidistantly spaced abscissas (43) one can write the option pricing Equation (42) for different strike values (44, 45) as a z-transform with $z_l = \mathrm{e}^{-\mathrm{i}\triangle v\triangle\kappa l}$:

$$C_{0l} = 2S_0 \mathrm{e}^{(\beta-1)\kappa_l - rT} \operatorname{Re} \sum_{k=0}^{n-1} \mathrm{e}^{\mathrm{i}\triangle v\triangle\kappa kl} a_j, \tag{46}$$

$$a_j = w_j \frac{\mathrm{e}^{\mathrm{i}j\triangle v\kappa_{\max}} \phi^Q(-\mathrm{i}v_j)}{2\pi v_j(v_j - 1)}$$

Setting

$$\triangle v\triangle\kappa = \frac{2\pi}{n} \tag{47}$$

Carr and Madan obtain a discrete Fourier transform in (46). Chourdakis [2004] points out that there is a fast algorithm for the z-transform that works even when $\Delta v\Delta\kappa \neq \frac{2\pi}{n}$ and $m \neq n$:

Chirp-z transform is an efficient algorithm for evaluating the z-transform for m different points z of the form

$$z_k = Aw^k,\ k = 0, 1, \ldots, m-1$$

where A and w are arbitrary complex numbers. The chirp-z transform works by rephrasing the original z-transform as a circular convolution and then computing this convolution by means of three FFTs as shown in Section I. For more details see Bluestein [1968], Rabiner, Schafer, and Rader [1969], and Bailey and Swartztrauber [1991]. Compared to the standard n-long FFT, the chirp-z algorithm is approximately $6(\ln m + 1)/\ln n$ times slower, for $m \leq n$.

The Matlab command for chirp-z transform of n-long input sequence a reads

$$\mathrm{czt}(a, m, w, A)$$

A GAUSS procedure czt.gss is available on my website.

The decision on whether to use the simple summation (42) m times, or whether to apply the chirp-z transform (46) depends on the desired number of strikes m. The speed of the former over the latter is roughly $m/6/(\log_2 m + 1)$ times higher. As a rough guide, for $m \approx 36$ the simple summation (42) is as fast as the chirp-z formula (46); for $m = 8$ it is three times faster; and for $m = 150$ it is three times slower.

One also has to decide whether to force the FFT spacing of strike values (47); this is done by boosting n while keeping Δv fixed. Suppose that $v_{\max}$ is chosen sufficiently high to achieve desired accuracy for a single strike. As a rule of thumb, if the initial spacing $2\pi/\operatorname{Im} v_{\max}$ is six times coarser than the desired spacing of log strikes $\Delta\kappa$, one should use the chirp-z transform; otherwise it will be faster to increase

n to satisfy (47) and use the short FFT algorithm described in Bailey and Schwarztrauber [2004, pp. 392-393].

The value of $\mathrm{Im}\,\nu_{\max}$ tends to be higher for short maturities and for parametric distributions with heavy tails, such as variance gamma or generalized hyperbolic. In such circumstances the FFT formulas in (46)–(47) are preferable. Non-parametric empirical equity return distributions have characteristic functions that decay faster, leading to lower values of $\mathrm{Im}\nu_{\max}$, leaving the chirp-z transform as the best option.

IV. CONCLUSIONS

There are three contributions in this article. I explain the working of the discrete Fourier transform in non-technical language using the familiar binomial option pricing model. Second, I highlight the common perils in computer implementation of fast DFT algorithms. Finally, I explain how the binomial pricing formula relates to more complex continuous-time models routinely used in the finance industry that allow for excess kurtosis, stochastic volatility, and leverage effects.

APPENDIX

Inverse Discrete Fourier Transform

To show $\mathcal{F}^{-1}[\mathcal{F}(a)] = a$, we need to prove that for $b = \mathcal{F}(a)$ defined in Equation (11) we have $\mathcal{F}^{-1}(b) = a$. Denote $\tilde{a} = \mathcal{F}^{-1}(b)$ and express $\tilde{a}$ from Equation (12):

$$\tilde{a}_l = \frac{1}{\sqrt{n}} \sum_{k=0}^{n-1} b_k z_n^{-kl}$$

Now substitute for b_k from (11):

$$\tilde{a}_l = \frac{1}{n} \sum_{k=0}^{n-1} \left(\sum_{j=0}^{n-1} a_j z_n^{jk} \right) z_n^{-kl}$$

Move z_n^{kl} inside the inner summation:

$$\tilde{a}_l = \frac{1}{n} \sum_{k=0}^{n-1} \left(\sum_{j=0}^{n-1} a_j z_n^{k(j-l)} \right)$$

Change the order of summation:

$$i_l = \frac{1}{n} \sum_{j=0}^{n-1} a_j \left(\sum_{k=0}^{n-1} \left(z_n^{j-l} \right)^k \right)$$

and take a_j in front of the inner summation (it does not depend on k):

$$\tilde{a}_l = \frac{1}{n} \sum_{j=0}^{n-1} a_j \left(\sum_{k=0}^{n-1} \left(z_n^{j-l} \right)^k \right)$$

By virtue of Equations (8) and (9), the inner summation $\sum_{k=0}^{n-1} \left(z_n^{j-l} \right)^k$ equals 0 for $j \neq l$ and n for $j = l$. Consequently:

$$\tilde{a}_l = \frac{1}{n} \sum_{j=0}^{n-1} a_j \left(\sum_{k=0}^{n-1} \left(z_n^{j-l} \right)^k \right) = a_l$$

for all l, which proves that $\mathcal{F}^{-1}(\mathcal{F}(a)) = a$.

Fourier Transform of Convolutions

We wish to show $\mathcal{F}(a \circledast b) = \sqrt{n}\mathcal{F}(a)\mathcal{F}(b)$. Let us begin by computing $c = a \circledast b$. From Equation (23):

$$c_j = \sum_{k=0}^{n-1} a_{j-k} b_k \qquad \text{(A-1)}$$

Denote by d the Fourier transform of c [$d = \mathcal{F}(a \circledast b)$] and use Equation (11) to evaluate d_l:

$$d_l = \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} c_j z_n^{jl}$$

Now substitute for c_j from Equation (A-1):

$$d_l = \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} \left(\sum_{k=0}^{n-1} a_{j-k} b_k \right) z_n^{jl}$$

Move z^{jl} inside the inner parentheses, writing it as a product $z^{jl} = z^{(j-k)l} z^{kl}$:

$$d_l = \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} \sum_{k=0}^{n-1} a_{j-k} z^{(j-k)l} b_k z^{kl}$$

Change the order of summation:

$$d_l = \frac{1}{\sqrt{n}} \sum_{k=0}^{n-1} \sum_{j=0}^{n-1} a_{j-k} z^{(j-k)l} b_k z^{kl}$$

and take $b_k z^{kl}$ infront of the inner summation (it does not depend on j):

$$d_l = \frac{1}{\sqrt{n}} \sum_{k=0}^{n-1} b_k z^{kl} \left(\sum_{j=0}^{n-1} a_{j-k} z^{(j-k)l} \right) \tag{A-2}$$

It is easy to see that the inner summation does not depend on k, because it always adds the same n elements. Only the order in which these elements are added depends on k (we are completing one full turn around the circle, starting at k-th spoke).

Hence we have:

$$\sum_{j=0}^{n-1} a_{j-k} z^{(j-k)l} = \sum_{j=0}^{n-1} a_j z^{jl} \text{ for all } k$$

Substituting this into (A-2) we obtain:

$$d_l = \sqrt{n} \underbrace{\left(\frac{1}{\sqrt{n}} \sum_{k=0}^{n-1} b_k z^{kl} \right)}_{\tilde{b}_l} \underbrace{\left(\frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} a_j z^{jl} \right)}_{\tilde{a}_l}$$

From the definition of the forward transform Equation (11), $\tilde{a} = \mathcal{F}(a)$ and $\tilde{b} = \mathcal{F}(b)$, which completes the proof.

ENDNOTES

The author thanks David Miles and Jonathan Wainwright for suggesting important clarifications in an early draft. Peter Carr and Sanjiv Das provided helpful comments and suggestions. This is an adapted version of Černý [2004, Chapter 7]. GAUSS is a trademark of Aptech Systems, Inc.; Matlab is a registered trademark of The MathWorks, Inc.

[1]GAUSS programs *Binomial.gss* and *DFT.gss* available from author's website.

[2]The fast Fourier transform does not appear in undergraduate textbooks on numerical mathematics. The most useful introductory references are web-based; see http://www.fftw.org/links.html, and in particular the online manual in Hey [1999]. An efficient implementation of FFT for all transform lengths is suggested in Frigo and Johnson [1998]; it is used in Matlab. Efficient implementation of a mixed 2,3,5-radix algorithm is due to Temperton [1992]; it is used in GAUSS. Duhamel and Vetterli [1990] provide an excellent survey of FFT algorithms.

[3]The prime factor algorithm (PFA) works faster because the factors 2, 3, and 5 have no common divisors; see Temperton [1992].

[4]GAUSS code *FFT.gss* available on author's website.

[5]For the Fourier transform to work, $C_T(\ln S_T) S_T^{-\beta}$ must be integrable as a function of $\ln S_T$. There are derivative securities, such as call and put options, where one needs to take $\beta \neq 0$ to insure integrability ($\beta > 1$ for the call, $\beta > 0$ for the put).

[6]Some unrestrictive technical conditions must hold to make sure that for the ψ given in Equation (39) Equation (38) holds for all values of $\ln S_T$. See Chandrasekharan [1989].

REFERENCES

Albanese, C., K. Jackson, and P. Wiberg. "A New Fourier Transform Algorithm for Value At Risk." *Quantitative Finance,* 4 (2004), pp. 328-338.

Andreas, A., B. Engelmann, P. Schwendner, and U. Wystup, "Fast Fourier Method for the Valuation of Options on Several Correlated Currencies." In J. Hakala and U. Wystup, eds. *Foreign Exchange Risk.* London: Risk Publications, 2002.

Bailey, D.H., and P.N. Swartztrauber. "The Fractional Fourier Transform and Applications." *SIAM Review,* Vol. 33, No. 3 (1991), pp. 389-404.

Benhamou, E. "Fast Fourier Transform for Discrete Asian Options." *Journal of Computational Finance,* Vol. 6, No. 1 (2002).

Bluestein, L.I. "A Linear Filtering Approach to the Computation of the Discrete Fourier Transform." *IEEE Northeast Electronics Research and Engineering Meeting,* 10 (1968), pp. 218-219.

Carr, P., and D.B. Madan. "Option Valuation Using the Fast Fourier Transform." *Journal of Computational Finance,* 2 (1999), pp. 61-73.

Carr, P., and L. Wu. "Time-changed Lévy Processes and Option Pricing." *Journal of Financial Economics,* Vol. 71, No. 1 (2004), pp. 113-141.

Černý, A. *Mathematical Techniques in Finance: Tools for Incomplete Markets.* Princeton: Princeton University Press, 2004.

——. "The Risk of Optimal, Continuously Rebalanced Hedging Strategies and its Efficient Evaluation via Fourier Transform." Technical Report, The Business School, Imperial College London, 2003.

Chandrasekharan, K. *Classical Fourier Transforms.* New York: Springer, 1989.

Chiarella, C., and N. El-Hassan. "Evaluation of Derivative Security Prices in the Heath-Jarrow-Morton Framework as Path Integrals Using Fast Fourier Transform Techniques." *Journal of Financial Engineering,* Vol. 6, No. 2 (1997), pp. 121-147.

Chourdakis, K. "Option Pricing Using the Fractional FFT." Working paper. Available at www.theponytail.net, 2004.

Dempster, M.A.H., and S.S.G. Hong. "Spread Option Valuation and the Fast Fourier Transform." In H. Geman, D. Madan, S.R. Pliska, and T. Vorst, eds, *Mathematical Finance—Bachelier Congress 2000*. Berlin: Springer, 2002, pp. 203-220.

Duffie, D., D. Filipović, and W. Schachermayer. "Affine Processes and Applications in Finance." *The Annals of Applied Probability*, Vol. 13, No. 3 (2003), pp. 984-1053.

Duffie, D., J. Pan, and K. Singleton. "Transform Analysis and Asset Pricing for Affine Jump Diffusions." *Econometrica*, 68 (2000), pp. 1343-1376.

Duhamel, P., and M. Vetterli. "Fast Fourier Transforms: A Tutorial Review and a State of the Art." *Signal Processing*, 19 (1990), pp. 259-299.

Eberlein, E., U. Keller, and K. Prause. "New Insights into Smile, Mispricing and Value at Risk: The Hyperbolic Model." *Journal of Business*, Vol. 71, No. 3 (1998), pp. 371-405.

Frigo, M., and S.G. Johnson. "FFTW: An Adaptive Software Architecture for the FFT." *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing* (1998), Vol. 3, pp. 1381-1384.

Heston, S. "A Closed-Form Solution for Options with Stochastic Volatility with Applications to Bond and Currency Options." *Review of Financial Studies*, 6 (1993), pp. 327-344.

Hey, A. "FFT Demystified." Available at http://www.eptools.com/tn/T0001/INDEX.HTM, 1999.

Hubalek, F., J. Kallsen, and L. Krawczyk. "Variance-Optimal Hedging and Markowitz-Efficient Portfolios for Processes with Stationary Independent Increments." Working paper, Technische Universität Wien, 2004.

Lee, R.W. "Option Pricing by Transform Methods: Extensions, Unification and Error Control." *Journal of Computational Finance*, Vol. 7, No. 3 (2004).

Madan, D., and E. Seneta. "The Variance Gamma Model for Stock Market Returns." *Journal of Business*, Vol. 63, No. 4 (1990), pp. 511-524.

Rabiner, L.R., R.W. Schafer, and C.M. Rader. "The Chirp Z-Transform Algorithm and its Application." *Bell Systems Technical Journal*, Vol. 48, No. 5 (1969), pp. 1249-1292.

Rebonato, R., and I. Cooper. "Coupling Backward Induction with Monte Carlo Simulations: A Fast Fourier Transform (FFT) Approach." *Applied Mathematical Finance*, Vol. 5, No. 2 (1998), pp. 131-141.

Temperton, C. "A Generalized Prime Factor FFT Algorithm for any n = 2p3q5s." *SIAM Journal on Scientific and Statistical Computing*, 13 (1992), pp. 676-686.

To order reprints of this article, please contact Ajani Malik at amalik@iijournals.com or 212-224-3205.

©Euromoney Institutional Investor PLC. This material must be used for the customer's internal business use only and a maximum of ten (10) hard copy print-outs may be made. No further copying or transmission of this material is allowed without the express permission of Euromoney Instituitonal Investor PLC. Copyright of Journal of Portfolio Management is the property of Euromoney Publications PLC and its content may not be copied or emailed to multiple sites or posted to a listserv without the copyright holder's express written permission. However, users may print, download, or email articles for individual use.