

Financial Market Simulation

In the 21st century.

Bruce I. Jacobs, Kenneth N. Levy, and Harry M. Markowitz

Large and detailed asynchronous (event-based) simulation models are widely used in the planning and analysis of systems such as manufacturing, logistics, and warfare. They have been used relatively little in financial analysis, especially as compared with continuous-time models. We believe the situation will change, starting early in the 21st century.

Here we describe asynchronous simulation generally, and illustrate its capabilities in terms of a particular asynchronous and discrete-event stock market simulator, the JLM Market Simulator (JLM Sim).

The JLM Sim is not a model of a market per se. Rather, it is a tool investors can use to create a model of the market using their own inputs. JLM Sim users can vary the number of securities, statisticians, portfolio analysts, investors, and traders in the simulated market and the decision rules they use. Prices are set endogenously, as new orders encounter already placed limit orders. At present, all JLM Sim investors are mean-variance investors; more advanced versions of JLM Sim later may accommodate additional investor types.

Dynamic models may be described as synchronous, asynchronous, or continuous. Asynchronous simulation has some advantages over synchronous or continuous-time financial models. We summarize the salient features of the current JLM Sim and some desirable features that may be added to more advanced versions.

TYPES OF DYNAMIC MODELS

Dynamic models can be divided between discrete and continuous. In discrete models, time advances in dis-

BRUCE JACOBS is principal at Jacobs Levy Equity Management in Florham Park, NJ.
bruce.jacobs@jacobslevy.com

KENNETH LEVY is principal at Jacobs Levy Equity Management in Florham Park, NJ.

HARRY MARKOWITZ is principal at Harry Markowitz Company in San Diego, CA.

crete increments, while in continuous models the system changes continuously over time. Discrete-time models can be broken down further into synchronous and asynchronous models. In synchronous-time models, the clock advances by fixed increments such as a day or a year. Typically, the status of all system constituents is updated at each increment of time.

In asynchronous models, time advances, typically by uneven increments, to the next scheduled event. The next event might be Investor A's placement of initial orders after reoptimization, followed perhaps a split second later by the review of an order placed previously by Trader B on behalf of Investor C, followed perhaps seconds later by the end-of-day event when accounts with short or leveraged positions are marked to market.¹

In an asynchronous model, events often involve only one or a few entities—one investor and a trader, or an investor selling and one or more buyers with orders on the books. An event may also involve many entities, as the end-of-day event does.

It is possible to solve some asynchronous-time models analytically; certain queuing models are a case in point (see Haverkort [1998] for examples on both sides). Most large and detailed asynchronous models that attempt to model some complex system fairly literally usually require computer simulation to derive their implications.²

JLM SIMULATOR

In a JLM Sim simulation run, as of any instant of simulated time, the simulated market has a state or *status*. This status changes at points in time called *events*.

Status

The status of the JLM Sim simulated market is described in terms of how many *entities* of different *entity types* there are; the values of their *attributes*; and the members of the *sets* they own. Entity types represented in JLM Sim include securities, investors, traders, order slips, portfolio analysts, and statisticians. A given simulation may include many individual entities of each type. Attributes include the price of a security and the volume traded so far today, the current wealth of an investor, and the buy or sell attributes of an order slip. As of any instant, an attribute of an (individual) entity has one and only one value.

Sets include the set of buy orders and the set of sell orders for a given security. The security's buy orders are sorted from high to low according to the limit price attribute of the order slip or, among order slips with the same limit price, according to an arrival time (actually, a take-a-num-

ber) attribute of the order slip. The sell order set is similarly sorted, but from low to high. We say that each security *owns* a buy-order set and a sell-order set. Zero, one, or more order slips *belong to*, or are *members of*, each set. (In a computer program, the sets would be named buy_orders and sell_orders sets, since in a computer program a hyphen (as in buy-orders) would be treated as a minus sign.)

Exhibit 1 summarizes some of the JLM Sim EAS (entity, attribute, and set) structure. We want to convey the general idea of the JLM Sim EAS structure, rather than provide a user's manual.

The entity types are listed first. The names of the attributes are in the second column, and the names of sets owned by the entities of the entity types are in the third. The last column, headed member or data type, shows such information as whether an attribute is an integer or real (decimal) number and what type of entity belongs to the named set.

The first entity type listed in Exhibit 1 is the system as a whole. Attributes of the system include the current lending and borrowing rates of interest. Sets owned by the system include all the securities, statisticians, and portfolio analysts in the simulation, as well as the kept trading days, kept months, investor templates, and trader templates.

Each simulated investor is created from some *investor template* and identified by the template it came from and a sequence number. For example, an investor could be the 456th investor from template 3. One attribute of an investor template is the number of investors that are to be generated from this template. One attribute of the investor is the template from which it comes.

All investors from a given template share certain attributes, such as reoptimization frequency and risk aversion parameter K. These are stored as attributes of the investor template. Investors from a given template differ with respect to attributes such as starting wealth and time at which they reoptimize. The starting wealth of an investor is drawn randomly at the beginning of the simulation from a lognormal distribution whose parameters are attributes of the investor template. Individual investors from a given template have differing experiences during a simulation run, depending in part on circumstances such as when they reoptimize and when their traders try to execute the resulting buy and sell orders.

When an investor wants to reoptimize, it chooses an "ideal" portfolio from a mean-variance efficient frontier that is based on estimates by one or another statistician. The choice of portfolio depends on the investor's risk aversion. The investor may seek to move only partway from current to ideal portfolio, depending on turnover constraints. To accomplish this, the investor places buy and sell orders with its trader.

Every trader is generated from a *trader template*. The number of traders generated from a given trader template is

EXHIBIT 1

Entity, Attribute, and Set Structure of JLM Sim (extract)

ENTITY TYPES	ATTRIBUTES	SETS OWNED	Member or Data Type
TheSystem	SimTime RFLendRatePerDay BrokerRatePerDay Liquidation_trader_nr		Real Real Real Integer
		Securities KeptTradingDays KeptMonths Statisticians PortfolioAnalysts InvestorsTemplates	Security Day Month Statistician PortfolioAnalyst Investor_template
Security	LastTradePrice Price StartOfDayPrice StartOfMonthPrice VolumeSoFarToday		Real Real Real Real Integer
		Buy_orders Sell_orders	Order Order
Security_X_Day	DailyReturn DailyVolume DailyClosePrice		Real Integer Real
Security_X_Month	MonthlyReturn MonthlyVolume MonthlyClosePrice		Real Integer Real
Statistician	EstMethodForMeans EstMethodForCovs		Enumeration Enumeration
Statistician_X_Security	AnnualizedMean		Real
Statistician_X_Security_X_Security	AnnualizedCov		Real
PortfolioAnalyst	StatisticianNr		Integer
		EfficientSet	EfficientSegment

not fixed. Rather, each investor template specifies, as an attribute, which trader template its investors will use. Traders from this trader template are created as needed to service investors who need to trade.

Attributes of a trader template include the alpha and the beta of the buy candidate (buy_alpha, buy_beta). To execute a buy order for a security, the trader initially places an order at a limit price, determined as follows:

$$\text{Limit Price} = \text{Buy-Alpha} + \text{Buy-Beta} \times \text{Price} \quad (1)$$

For example, if buy-alpha equals -0.02 and buy-beta equals 1.0, the trader bids two cents less than current price.

If buy-alpha equals 0.0 and buy-beta equals 1.01, the trader bids 1 percent more than current price (unless an offer is available at a lower price). Here “price” may equal the average of bid and asked prices, the bid or the asked price, or the last transaction price, depending on the availability of these prices and certain relationships among them.

If the security’s sell-order set includes a sell order at the buyer’s limit price or less, a transaction takes place for the lesser of the buyer’s desired trade size (amount_to_do) and the seller’s desired trade size (amount_to_do), where the “amounts to do” are attributes of the order slip entity. If this trade does not complete the buyer’s order, the amount to do on the buyer’s order slip is reduced by the amount of the security pur-

EXHIBIT 1 (continued)

Entity, Attribute, and Set Structure of JLM Sim (extract)

ENTITY TYPES	ATTRIBUTES	SETS OWNED	Member or Data Type
EfficientSegment	HighE		Real
	HighV		Real
	LowE		Real
	LowV		Real
	HighPortfolio		CornerPortfolio
	LowPortfolio		CornerPortfolio
CornerPortfolio	Cp_nr		Integer
	E		Real
	V		Real
Corner_Portfolio_X_Security	X		Real
InvestorTemplate	Nr_investors		Integer
	Portfolio_analyst_nr		Integer
	Trader_template_nr		Integer
	Mean_log10_init_wealth		Real
	Sigma_log10_init_wealth		Real
	K		Real
	Reoptimization_frequency		Enumeration
		Investors	Investor
InvestorTemplate_X_Security	Total_bought_today		Integer
	Nr_of_buyers		Integer
	Seq_nr_of_largest_buyer		Integer
	Purchase_of_largest_buyer		Integer
	Total_sold_today		Integer
	Nr_of_sellers		Integer
	Seq_nr_of_largest_seller		Integer
	Sale_of_largest_seller		Integer
Investor	Seq_nr		Integer
	Investor_template_nr		Integer
	StartingWealth		Real
	Deposits_received		Real
	Withdrawals_paid		Real
	Withdrawals_owed		Real
	Collateral_for_short_positons		Real
	CurrentWealth		Real

chased, and the process is repeated. If the buyer's remaining order cannot be filled by orders from the sell-orders set, it is entered into the buy-orders set.

The buyer waits a specified time (*buy_first_time_wait*) before attempting to complete the order by raising its bid. It adds specified increments to alpha and beta (*buy_alpha_inc*, *buy_beta_inc*), and recomputes a new limit price using Equation (1) but substituting the old limit price for price. If this does not result in a purchase, the trader waits a further specified time (*buy_following_time_wait*) before again recomputing the limit price.

This process is repeated a specified number of times

(*buy_max_nr_price_changes*). The buyer then waits a specified time (*buy_last_time_wait*) before canceling any uncompleted order. A similar procedure applies for sell orders using the attributes of the trader template.

Entity types listed in Exhibit 1 include "security_X_day," "security_X_month," "statistician_X_security," and so on. These are called *compound entity types*. "Investor_X_security," for example, is an investor-security combination. A "statistician_X_security_X_security" is a statistician-security-security triplet. Compound entities can have attributes and own sets. For example, "buy_or_sell_amount" is an attribute of "trader_X_security"; that is, each trader-

EXHIBIT 1 (continued)
Entity, Attribute, and Set Structure of JLM Sim (extract)

ENTITY TYPES	ATTRIBUTES	SETS OWNED	Member or Data Type
Investor_X_Security	X_units		Real
Trader_template	Buy_Alpha Buy_Beta Buy_Alpha_inc Buy_Beta_inc Buy_First_time_wait Buy_Following_time_wait Buy_Last_time_wait Buy_Max_nr_price_changes Sell_Alpha Sell_Beta Sell_Alpha_inc Sell_Beta_inc Sell_First_time_wait Sell_Following_time_wait Sell_Last_time_wait Sell_Max_nr_price_changes		Real Real Real Real Real Real Real Integer Real Real Real Real Real Real Real Integer
Trader	Trader_template_nr Investor_being_served		Integer Investor_ID
Trader_X_Security	Buy_or_sell_amount Amount_on_order	Orders_against_amount	Integer Integer Order_slip
Order_slip	Buy_or_sell Trader_placing_order Security_ordered Limit_price Amount_to_do Order_status		Enumeration Trader_ID Integer Real Integer Enumeration

security pair has a negative, zero, or positive amount associated with it, which indicates the number of units (shares or bonds) that are to be sold (if negative) or bought (if positive). “Amount_on_order” is another attribute of trader-security. The amount on order may be less than the buy or sell amount, because buy orders are sometimes delayed until cash is raised from sell orders.

“X_units” is the number of units (shares of stock or face value of bonds) attribute of investor-security; “estimated_annualized_mean” is an attribute of statistician-security; “estimated_annualized_cov(ariance)” is an attribute of statistician-security-security.

“Day” is an entity type in JLM Sim that has no attributes on its own, but appears in compound entity types. Specifically, “daily return,” “daily volume,” and “daily close price” are attributes of security-day. The system owns sets of recent “kept days” and “kept months” that provide data needed by statisticians or for other purposes.

In general, then, as of any instance in simulated time

in a JLM Sim run, the status of the system is described by the values of the attributes and members of sets owned by individuals of entity types (including compound entity types) such as those listed in Exhibit 1.

Events

Events change the status of the simulation and cause future event occurrences. Exhibit 2 lists the principal event types of JLM Sim and the actions they bring about. The initialization routines are not events per se, but are included in the exhibit because they change the status of the simulation (from nothing to something) and cause the first event occurrences.³

Initialization. The initialization routines create statisticians, portfolio analysts, investor templates, and trader templates with the attributes specified by the user. In the current version of JLM Sim, statisticians use historical returns to estimate covariances, and, depending on the expected return

estimation procedure specified, may use historical returns to estimate expected returns. The returns the statisticians find in *kept days* and *kept months* at the start of the simulation are randomly generated by a factor model specified by the user.

The number of investors to be generated from a given investor template is an attribute specified by the JLM Sim user. When each investor is created during initialization, its starting wealth is drawn randomly from a lognormal distribution whose user-specified parameters are attributes of the investor template.

Another attribute governs whether investors from a given investor template reoptimize daily, monthly, quarterly, or annually. If investors from an investor template reoptimize monthly, for example, an initialization routine determines when during the first month of the simulation a particular investor will first reoptimize.⁴

Reoptimization. The reoptimization event cancels any buy or sell orders the investor has outstanding. The investor's portfolio analyst selects an ideal portfolio that, if there were no transaction costs, would maximize:

$$(\text{Portfolio Expected Return}) - K(\text{Portfolio Variance}) \quad (2)$$

where K , the investor's risk aversion, is an attribute of the investor template. The investor computes the buy and sell orders needed to move the portfolio toward the ideal portfolio. The amount of movement from current toward ideal can be regulated by constraints intended to reduce costly turnover.

Desired purchases and sales are handed to the trader to execute as we have described. The trader first attempts to execute each order to sell long positions or to cover short positions. Before placing an order to buy or to sell short, the trader considers the possibility that the purchase or short sale will be executed before the long sale or the short cover, putting the investor in violation of some regulation or self-imposed investor constraint. In that case, some or all purchases or short sales may be postponed until sufficient long sales or short covering transactions are completed.

When an order is placed, JLM Sim determines whether there is an order on the books at the limit price or better. If there is, a transaction takes place. If the transaction does not complete the new order, further transactions are sought from the book; any uncompleted portion is finally "placed on the books"—entered into the set of buy orders or sell orders—and an order review is scheduled. If, before this time, the order is filled by a transaction emanating from some other investor's reoptimization or order review, the order review is cancelled.

Order Review. In the order review event, the order is either repriced or cancelled. If the order is repriced, JLM Sim considers whether there are one or more matching orders. If the order is not filled, it is placed on the books again, and

EXHIBIT 2

JLM Sim Events

Event	Actions
Initialize	Creates and initializes status in accord with the JLM Sim user's instructions.
Reoptimize	May randomly generate deposits and withdrawals for the particular investor. Has investor's portfolio analyst compute the investor's ideal portfolio, using estimates supplied by its statistician. Investor computes how far it should move from its current portfolio toward the ideal portfolio, considering turnover constraints. Places orders with trader. Trader executes orders if there are matching orders on other side, and places balance of order on books. If trades are executed, trader for the other party may take further actions.
Review Order	Changes the limit price or cancels the order. If limit price is changed, actions may occur similar to those that occur when an order is placed during reoptimization.
End of Day	Updates daily and perhaps monthly statistics. Marks to market accounts with leverage or short positions. Accounts that violate maintenance margin requirements are turned over to a liquidation trader.

a new order review is scheduled. If a transaction takes place, and the matching investor has raised cash by selling or has covered a short position, then the matching investor may take further action.

End of Day. The end-of-day event updates daily statistics and perhaps monthly statistics. It checks to see if accounts with leverage or short sales have violated maintenance margin requirements. Any account that is found in violation of these requirements is turned over to a liquidation trader, who trades to get the account back into compliance.

One trader template is designated by the JLM Sim user as the liquidation trader template (an attribute of the system). Liquidation traders are created as needed, and use the liquidation trader template's parameters (such as sell-alpha and sell-beta). The parameters of the liquidation trader template, specified by the JLM Sim user, are presumably more aggressive than those of other traders, in that they are slanted toward quick execution rather than favorable prices.

OBJECTIVES AND EXTENSIONS

The JLM Market Simulator is not a model of a market per se; rather, it is a tool that allows its user to model a market by supplying certain components. When JLM Sim users specify the numbers of entities of different types and their attributes, they have in effect created a model of a market. This model is run for the length of time specified by the user—and can be run repeatedly with different initial random number seeds—in order to estimate the probable outcomes.

At first, the user will need to experiment with the model to get it to reflect aspects of a real-world market. Not only should the components of the model imitate, to some extent, their real-world counterparts, but the resulting price behavior should be reasonably comparable to its real-world counterpart. When the model has achieved some plausibility in terms of inputs and outputs, it can be used to test investment and trading policies, or regulatory policies such as the efficacy of the uptick rule and the relationship between changes in interest rates and market response.

We expect to enhance JLM Sim, in part based on the experience of users of the current version. Below are two areas in which enhancements seem attractive.

Alternative Investor and Trader Behaviors

In the current JLM Sim, all investors are mean-variance investors. Other investors may use other criteria and procedures, such as downside risk, mean-absolute deviation, or resampled frontiers. Behavioral finance specialists have described non-rational market behavior. We would like to make any and all such investor behaviors available to the JLM Sim modeler.⁵

One way to do so is for us to program such alternative investor behaviors into JLM Sim and let the user specify which behavior is to be used by investors of a given investor template. The advantage of this approach is that it puts the given behavior at the user's disposal without requiring that the user program it. The disadvantage is that the user is limited to the behaviors the originators programmed.

Another approach would be to allow users to program their own proposals for investor behavior. The most natural way to do this would be for the user or user's programmer to program in C++, the computer language in which JLM Sim is programmed. This would make use of C++'s ability to have one version of an object (entity) extend or override another version of the object, making use of public information about other entities (such as the bid and asked prices for a stock) without being able to directly modify such information.

The advantage of this approach is that it is open-ended; the disadvantage is that it requires the user to be or have access to a C++ programmer.

Model Size

JLM Sim runs fairly fast, at a few thousand events per second, on a 2.4 GHz PC, primarily for two reasons. One reason is that JLM Sim stores the simulated status (entities, attributes, and sets) and the calendar of coming event notices in the computer's random access memory rather than the longer-to-access disk memory. The second reason is that JLM Sim uses a particular software to file, remove, and find members of very large, ordered sets, such as the set of coming events, that currently handles only sets stored in RAM.⁶

For these two reasons, a JLM Sim simulation run on a personal computer must fit into the PC's virtual RAM, and will run especially fast if it fits into the PC's real RAM for the most part. As a rough rule of thumb, this sets an upper limit for JLM Sim at roughly a few tens of thousands of investors if there are only a few securities (say, ten or fewer), or at fewer investors if there are many securities. These limits should be sufficient for experimenting with markets that have features of real markets, but are smaller.

For example, in our own tests of JLM Sim features, we think of a run's thousands of investors as thousands of investment companies with random deposits and withdrawals rather than as a market of individuals.

If we tried to build a life-sized market on a PC by using disk memory as if it were a large RAM, we would slow the simulator by orders of magnitude because of the orders-of-magnitude difference between the access times of RAM and disk. We believe this size bottleneck can be overcome, and that life-sized markets can be simulated economically with the hardware currently available, but programming beyond that of JLM Sim will be required.

Part of the solution to the size bottleneck is to keep on disk only, not in RAM, data that probably won't be needed for some time, such as information about investors who optimize monthly or quarterly when it is not their time to reoptimize. Thus RAM would be used as a large cache.

Another part of the solution is to use multiple PCs, perhaps linked by the Internet. Such a distributed simulation is not uncommon now (see Fujimoto [1998]). Market simulation should be well suited to this mode of computing, because of the intrinsically decentralized nature of much market activity.

ADVANTAGES OF ASYNCHRONOUS FINANCE MODELS

The most common dynamic financial models assume that security prices follow a continuous-time process, which is either a Brownian motion or a function of a Brownian motion. Physicists, too, are sometimes content to describe the movements of particles suspended in liquid as a Brownian motion or similar continuous random path. For other purposes, however, they look behind the scenes at the molecules that jostle

the particles and cause their erratic motion, or the atoms that bind together to form molecules, or the electrons, protons, and neutrons that form atoms, or the quarks that constitute the neutrons and protons.

In finance as well, it may be sufficient for some purposes to assume that prices follow a given random process. For other purposes, it is essential to look behind the scenes at what causes price movements.

A major advantage of continuous-time models that represent price movements by given random processes is that some of them can be solved explicitly. This allows the analytic evaluation of investment strategies, or of the values of prices derived from the given price series, such as the price of an option, given the price process of the underlying security. But the assumption of a given and fixed price process is sometimes questionable.

For example, investment actions may change the price process, or a change in the composition of the behind-the-scenes agents may change the price process. Continuous-time models may also be insufficient when the question to be analyzed is whether micro theories about the behavior of investors add up to the observed macro phenomena of the market.

Consider liquidity, which has proved to be a problem for large investors. One extreme case is Black Monday, October 19, 1987, which many believe was exacerbated by an option-based strategy known as portfolio insurance. Other extreme cases include the collapse of hedge funds that have found the liquidity of their positions drying up just when they needed to liquidate them.⁷

In these and in many less extreme cases, investment policies should be evaluated taking into account the fact that the large investor is not a price-taker; rather its own actions affect the price process. Our view is that an asynchronous market simulator such as JLM Sim is best equipped to handle this by representing the agents and market mechanisms behind the observed prices.

Kim and Markowitz [1989] present an asynchronous simulation whose investors are either rebalancers or (constant-proportion) portfolio insurers. They show that the behavior of the market changes radically as the proportion of one kind of investor varies in relation to another. Similarly, debugging runs during the development of JLM Sim show that the behavior of the market varies with the composition of the mean-variance investors in the market, depending on their estimation procedures and risk aversion.

Prior to the 1987 market break, there was an increasing use of portfolio insurance. Perhaps a model that incorporated the actual procedures of the parties in the market, portfolio insurers and others, could have anticipated the consequences of this shifting composition of market participants. More generally, it seems to us that, in order to predict the consequences of future trends in the composition of

the market, one needs a model that reflects the differing procedures of different kinds of participants. Asynchronous simulation is well suited to this purpose.

Various hypotheses have been put forth as to the behavior of individual or institutional investors. Some of these hypotheses postulate optimizing behavior of one sort or another. Other hypotheses postulate not-necessarily-rational investor reactions to events. Both the optimizing and the behavioral hypotheses admit different versions, and the market surely includes a mixture of investors with differing investment patterns. One function of a detailed asynchronous simulation is to determine the consequences of a proposed population of investors characterized by one or more behavior patterns, to see whether the postulated behavior patterns add up to observed market processes. A model that starts by *assuming* some random price process cannot *deduce* this process from investor characteristics.

An alternative method of dynamic modeling is synchronous discrete, as opposed to continuous or asynchronous discrete modeling. For example, the microscopic simulation model of Levy, Levy, and Solomon [2000] is a discrete synchronous model. In each period, a market equilibrium price is computed from the demand and supply curves of all investors. These demand curves are based on optimizing or behavioral considerations similar to those that can be incorporated into an asynchronous model.

But a world where prices are set by an equilibrium calculation based on all investors' demand and supply curves is a world different from one where investors can enter and leave the market at any time and may or may not find other investors waiting for them. When we look at actual markets, we see the latter. We contend, therefore, that a model such as the JLM Sim is capable of a more literal representation of the world than the LLS microscopic simulation model.

More generally, asynchronous simulation, allowing action at any time by anyone, is capable of a more literal representation of actual markets than discrete synchronous models, which typically have to make some special assumption about the effect of everyone acting at the same time at assumed discrete times.

CAVEAT

Before we can use a model for prediction and policy evaluation, we must verify that its implications approximate reality under circumstances observed in the past. We should not expect it to be an easy task to build a complex asynchronous simulation with reasonably realistic properties. A lesson we learned from debugging runs of JLM Sim illustrates this.

In early runs, with two securities and 4,000 investors, we found that the price of stocks doubled and redoubled or fell by comparable amounts in the course of a day, even in a market with no news. Since these stocks were substantially

priced, not penny stocks, such frequent explosive price movements were obviously unrealistic.

One problem, we found, was that portfolio analysis that uses historical averages for expected returns is not sensitive to large short-term changes in price. A second problem was that the simulated traders (following our bidding rules) had no sense of recent price levels.

Of our 4,000 investors in these runs, 1,000 each of two different investor templates reoptimized daily. Suppose that, on a particular day, investors of one of these daily reoptimizing templates are inclined to buy security A. At various times during the day, individual investors of this template would instruct their traders to purchase the desired numbers of shares. In some cases, the traders' decision rule was to bid slightly higher than the current price (which may be the average of the bid and ask, or possibly just the bid if there is no ask). Once the supply offered on the books was used up, one trader raised the current bid a bit, and the next a bit more, the next a bit more, unmindful of the fact the stock that they had just bid \$200 for sold for \$100 earlier in the day.

Part of the cure for the explosive consequences of the initial models was to allow the JLM Sim user to specify "anchoring rules" for traders. For example, anchoring rules provided in JLM Sim instruct the trader to bid as we have described, but to not bid more than the average or the maximum of the recent closing price plus some percentage or plus some number of standard deviations, and, similarly, not to offer less than the average or minimum of the recent closing price minus some percentage or some number of standard deviations.

The JLM Sim modeler can specify different anchoring rules or different parameter values, such as the number of days to be used in computing recent minimums, maximums, or averages, for different trader templates. These anchoring rules help eliminate the problem of frequent explosions and implosions.

The idea that traders have some sense of how much to pay for a security, and that this price is somehow related to the prices at which the security has recently traded, is perhaps too obvious to mention. But if you do not tell this to the model, the model does not know.

What else of importance have we forgotten to tell the model? Inevitably, many things.

Whether we have left out something essential will be determined, in large part, by comparing the implications of the model with the behavior of actual markets. This is an iterative process. When we have a model that imitates the coarse features of the world, then we seek one that imitates its finer features. At some point in this process, we are willing to give some weight to the model's answers to questions such as, within the world as modeled, what are better and worse policies, and what are the effects of changing conditions.

CONCLUSION

Asynchronous-time, discrete-event simulation is widely used to model complex systems, but has seldom been used to model financial markets. We have sketched out the JLM Sim to illustrate the nature and capabilities of asynchronous market simulation.

The status of a JLM Sim simulation is given by the attribute values and set memberships of entities such as investors, traders, portfolio analysts, statisticians, and securities. Events such as reoptimization, order review, and end of day change the status of the simulation and cause subsequent event occurrences. In an asynchronous simulation, time advances to the next most imminent event, typically in varying increments.

Certain continuous-time financial models, which assume particular price processes, have the advantage of being analytically solvable, but such models are inappropriate when the policies to be evaluated change the price process, perhaps in some non-obvious way; the changing composition of market participants changes the price process; or the issue at hand is whether the postulated micro behavior of financial agents, plus market mechanisms, implies observed market macro behavior. Asynchronous simulation is well suited to such analyses.

ENDNOTES

¹Synchronous versus asynchronous simulation is sometimes referred to as time-based versus event-based simulation (see Haverkort [1998, pp. 412-417]), or as fixed-increment time advance versus next-event time advance (see Law and Kelton [2000]). For examples of applications of simulation methodology, see Banks [1998], particularly the articles on manufacturing simulation (Rohrer [1998] and Ulgen and Gunal [1998]); logistics and transportation systems simulation (Manivannan [1998]); computer and communications systems simulation (Hartmann and Schwetman [1998]); and military simulation (Kang and Roland [1998]).

Discrete-event simulations of financial markets are surveyed by Levy, Levy, and Solomon [2000]. These are mostly synchronous simulation models. Continuous-time models in finance are surveyed in Merton [1990].

The JLM Market Simulator will soon be available, gratis, on the web. See www.jacobslevy.com for availability announcements.

²See Jacobs and Levy [1989] for a discussion of a taxonomy used in the sciences to classify systems into three types—ordered, complex, and random—and its application to the stock market.

³Simulation programming languages such as SIMULA (see Dahl and Nygaard [1966]) and SIMSCRIPT II.5 [1987] describe status change in terms of processes instead of or in addition to event occurrences. That is, instead of an order review event, we could speak of a trading process that includes a wait statement between the order placing action and the order review action. This process view is often helpful in modeling. At the implementation level, however, the simulation is in fact an asynchronous event simula-

tion; end-of-wait coming-event notices are placed on the calendar like events in an event-oriented simulation. In the case of JLM Sim, which is implemented in C++ using the EAS-E.org package for handling large ranked sets, this is easiest to program as an event-oriented simulator.

⁴What actually happens within the computer is that a reoptimization coming-event notice is filed into the calendar, which is a set of coming events ordered by time of occurrence. After initialization is completed, the timing routine is called upon. The timing routine removes the first, most imminent, event from the calendar, calls upon the appropriate event routine, and hands the coming-event notice to the event routine. When the reoptimization coming-event notice for a particular investor is removed from the front of the calendar set, the reoptimization event is called, and is given the event-notice that remembers the investor to be reoptimized.

⁵Mean-absolute deviation is discussed in Konno and Yamazaki [1991]. The resampled frontier is discussed in Michaud [1998] and Markowitz and Usmen [2003]. Downside risk, also known as semi-deviation, is the subject of an issue of *The Journal of Investing* [1994]. Behavioral finance is surveyed in Shefrin [2001].

⁶The EAS-E software is available free at www.eas-e.org.

⁷See Jacobs [1999, 2004], Jacobs and Levy [2005], Kim and Markowitz [1989], and *Report of the Presidential Task Force* [1988].

REFERENCES

- Banks, Jerry, ed. *Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice*. New York: John Wiley & Sons, 1998.
- Dahl, O., and K. Nygaard. "SIMULA—an ALGOL-Based Simulation Language." *Comm. ACM*, 9, No. 9 (September 1966), pp. 671-678.
- Fujimoto, Richard M. "Parallel and Distributed Simulation." Chapter 12 in Banks [1998], pp. 429-464.
- Hartmann, Alfred, and Herb Schwetman. "Discrete-Event Simulation of Computer and Communication Systems." Chapter 20 in Banks [1998], pp. 659-676.
- Haverkort, Boudewijn R. *Performance of Computer Communication Systems: A Model-Based Approach*. New York: John Wiley & Sons, 1998.
- Jacobs, Bruce I. *Capital Ideas and Market Realities: Option Replication, Investor Behavior, and Stock Market Crashes*. Oxford: Blackwell, 1999.
- . "Risk Avoidance and Market Fragility." *Financial Analysts Journal*, January/February 2004.
- Jacobs, Bruce I., and Kenneth N. Levy. "The Complexity of the Stock Market." *The Journal of Portfolio Management*, 16, No. 1 (Fall 1989), pp. 19-27.
- . "A Tale of Two Hedge Funds." In Bruce I. Jacobs and Kenneth N. Levy, eds., *Market Neutral Strategies*. New York: John Wiley & Sons, 2005.
- Kang, Keebon, and Ronald J. Roland. "Military Simulation Systems." Chapter 19 in Banks [1998], pp. 645-658.
- Kim, G., and Harry M. Markowitz. "Investment Rules, Margin and Market Volatility." *The Journal of Portfolio Management*, 16, No. 1 (Fall 1989), pp. 45-52.
- Konno, H., and H. Yamazaki. "Mean-Absolute Deviation Portfolio Optimization Model and its Applications to Tokyo Stock Market." *Management Science*, 37, No. 5 (May 1991).
- Law, Averill M., and W. David Kelton. *Simulation Modeling and Analysis*, 3rd ed. New York: McGraw-Hill, 2000.
- Levy, Moshe, Haim Levy, and Sorin Solomon. *Microscopic Simulation of Financial Markets: From Investor Behavior to Market Phenomena*. Berkeley, CA: Academic Press, 2000.
- Manivannan, Mani S. "Simulation of Logistics and Transportation Systems." Chapter 16 in Banks [1998], pp. 571-604.
- Markowitz, Harry M., and Nilufer Usmen. "Resampled Frontiers versus Diffuse Bayes: An Experiment." *Journal of Investment Management*, 4, No. 1 (2003), pp. 9-25.
- Merton, Robert C. *Continuous-Time Finance*. Cambridge: Blackwell, 1990.
- Michaud, Richard O. *Efficient Asset Management: A Practical Guide to Stock Portfolio Optimization and Asset Allocation*. Boston: Harvard Business School Press, 1998.
- Report of the Presidential Task Force on Market Mechanisms* (The Brady Commission). Government Printing Office, January 1988.
- Rohrer, Matthew W. "Simulation of Manufacturing and Material Handling Systems." Chapter 14 in Banks [1998], pp. 519-545.
- Shefrin, Hersh, ed. *Behavioral Finance*, Vol. 1. Northampton, MA: Edward Elgar, 2001.
- SIMSCRIPT II.5 Programming Language. CACI Products Company, 1987.
- Ulgen, Onur, and Ali Gunal. "Simulation in the Automobile Industry." Chapter 15 in Banks [1998], pp. 547-570.
- To order reprints of this article, please contact Ajani Malik at amalik@ijjournals.com or 212-224-3205.*
- Reprinted with permission from the *The Journal of Portfolio Management 30th Anniversary Issue*. Copyright 2004 by Institutional Investor Journals, Inc. All rights reserved. For more information call (212) 224-3066. Visit our website at www.ijjournals.com.