Efficient Portfolios, Sparse Matrices, and Entities: A Retrospective

Harry M. Markowitz,

EFFICIENT PORTFOLIOS, SPARSE MATRICES, AND ENTITIES: A RETROSPECTIVE

HARRY M. MARKOWITZ

1010 Turquoise Street, Suite 245, San Diego, California 92109

In 1989 I was pleased and honored to be awarded the ORSA/TIMS (now INFORMS) John von Neumann Theory Prize for my work in portfolio theory, sparse matrices, and SIMSCRIPT. The following is a retrospective on my work in these fields.

PORTFOLIO THEORY

The Epiphany. One day in 1950, in the library of the Business School of the University of Chicago, I was checking out the possibility of writing my Ph.D. dissertation for the Economics Department in some area which applied mathematical or statistical techniques to "the stock market." My adviser, Professor Jacob Marschak, had sent me to Professor Marshall Ketchum in the Business School for a reading list. I had taken no courses in finance but, following Professor Ketchum's reading list, I had worked my way through Graham and Dodd's (1951) *Security Analysis*, had examined Wiesenberger's (1941) survey of *Investment Companies*, and now began reading John Burr Williams (1938) *The Theory of Investment Value*.

Williams asserted that the value of a stock is the expected present value of its future dividends. It struck me that if the investor, or investing institution, was concerned only with the expected value of a stock, it must also be only concerned with the expected value of its portfolio-as-a-whole. But maximizing the expected value of a portfolio requires investment in only one stock. Diversification is a common practice, as I saw in Wiesenberger's survey of investment companies, and makes sense since, as everyone knew even then, one should "not put all one's eggs in one basket." Clearly, investors diversify to avoid risk. What was missing from Williams' analysis was the notion of the risk of the portfolio-as-a-whole. An obvious measure which came to mind was variance or, equivalently, standard deviation. A check with a probability book on the library shelf showed that the variance of a weighted sum of random variables (e.g., the return on a portfolio of securities) involves the covariances or correlations among the random variables, as well as the weights and individual variances. This made good sense. It said that the riskiness of the portfolio had to do not only with the riskiness of the individual securities therein, but also to what extent they moved up and down together.

Since there were two quantities involved, I drew a graph with risk (standard deviation) on one axis and return (i.e., expected return) on the other. I was currently taking T. J. Koopmans' course on "activity analysis" in which he distinguished between efficient and inefficient combinations of production activities (Koopmans 1951). I accordingly labeled as efficient those combinations of risk and return which were not dominated by some other such combination.

The above thought process occurred while I was still in the middle of reading the Williams' book. Later in the book Williams deals with the question of risk essentially by advising the investor to invest in a large number of securities, presumably all with roughly maximum expected return. I felt that the theory which I had now sketched was more satisfactory than that which Williams presented, since he made no mention of avoiding covariance, combining securities efficiently into portfolios, or the risk-return trade-off of the portfolio-as-a-whole. I proposed to the Economics Department that I write a dissertation on "portfolio selection," and this was accepted.

The inputs to a portfolio analysis consist of the means and variances of individual securities and correlations between these securities. The outputs consist of efficient risk-return combinations and the portfolios which give rise to them. I assumed that it was the job of the security analyst to provide the inputs to the portfolio analysis. I took it as my task to figure out how to derive efficient sets from these inputs. It turns out that the set of efficient portfolios is piecewise linear, consisting of a finite number of pieces. Thus the entire efficient frontier can be written down by characterizing these finite number of pieces. The critical line algorithm for tracing out the efficient frontier is presented in Markowitz (1956). The basic tool used is the Kuhn-Tucker (1951) theorem. Markowitz (1959) Appendix A, shows that the algorithm works even if the covariance matrix is singular. Markowitz (1952) presented a geometric description of the set of efficient portfolios. Perold (1984), Markowitz (1987), and Markowitz and Todd (2000) discuss the shapes, properties, and computation of mean-variance efficient sets.

Subject classifications: Finance, portfolio: origins of portfolio theory. Professional: comments on. Programming, linear: sparse matrices. Simulation, languages: SIMSCRIPT.
Area of review: Anniversary Issue (Special).

Dantzig's Influence. I left Chicago in the Fall of 1951 for the RAND Corporation in Santa Monica with my course work finished but my dissertation still to be written. It was my good fortune to have George Dantzig join RAND about a year after I got there. George was the focal point of those doing work on linear programming. In the next section I will discuss my one noteworthy contribution to this area. In this subsection I note ways in which Dantzig's work contributed to the solution of the portfolio selection problem.

Markowitz (1956) defines the portfolio selection problem as that of finding mean-variance efficient portfolios subject to linear equality and inequality constraints. This is the same constraint set as that of linear programming, but with mean-variance efficiency rather than the optimization of a linear function as the objective. The portfolio with maximum expected return, when it exists, is the natural starting point in tracing out the set of efficient portfolios. Since expected return is a linear function of portfolio investments, finding the portfolio with maximum expected return is a linear-programming problem. Dantzig's simplex algorithm not only provides the solution to this problem, but also provides the critical line algorithm with various other services. In particular, it determines whether or not the constraint set specified by the analyst is feasible, whether or not feasible portfolio expected return is bounded and, if the model is rank deficient, it provides an equivalent model which is not rank deficient.

Along any piece of the set of efficient portfolios, security holdings X_i, $i = 1, \ldots, n$, vary linearly. Also the η_i (the partial of a Lagrangian with respect to the X_i) vary linearly. The algorithm knows when a corner portfolio is reached, where one efficient segment of the efficient set meets the next segment of the efficient set, by determining which event happens first: an $X_i \downarrow 0$ for an IN variable or an $\eta_i \downarrow 0$ for an OUT variable. (IN variables may vary on a critical line; OUT variables may not.) But what should be done if two or more of these variables go to zero simultaneously? Usually it is sufficient to break ties arbitrarily, but conceivably this could get one into trouble. The solution to this problem presented in Markowitz (1956) is an adaptation of the Dantzig solution to a comparable problem for the simplex algorithm.

Wolfe's Generalization. My work at the RAND Corporation did not include "portfolio analysis." But no one objected to my taking the time to write my 1952 and 1956 articles. I submitted the latter to the *Naval Research Logistics Quarterly* edited by Alan J. Hoffman. Elsewhere, Phil Wolfe had been working on the quadratic-programming problem, to minimize a quadratic function ($\text{Q} - \lambda\text{L}$, Q positive semidefinite, L linear) subject to linear constraints. Wolfe also submitted his work to *NRLQ*. Hoffman sent Wolfe's paper to me and my paper to Wolfe for refereeing. We both recommended that the other paper be published, and both were. As a by-product of tracing out the efficient frontier, the critical line algorithm minimizes $\text{Q} - \lambda\text{L}$ (for variance Q and expected return L) for all $\lambda \geqslant 0$. Thus the critical line algorithm is , incidentally, a quadratic-programming algorithm. It struck Phil Wolfe that the critical line algorithm solves the quadratic-programming problem in a sequence of steps which are precisely the same as the steps by which the simplex algorithm solves the linear-programming problem, with one exception. The variables of the quadratic program come in pairs X_i, η_i. When one of these pairs is IN the linear programming "basis," the other is OUT. Wolfe thus defined quadratic programming as an example of linear complimentarity programming. At first it seemed that the practical use of this observation was to easily convert a linear-programming code into a quadratic-programming (or portfolio selection) code. Subsequently, it was found that other problems satisfied the linear complimentarity format, e.g., non-zero-sum games (Lemke 1965).

Tobin's Invitation. While at the University of Chicago, I was a student member of the Cowles Commission for Research in Economics under the guidance of its head, Tjalling Koopmans, and its former head Jacob Marschak. Former members of this relatively small organization have included Nobel Laureates Kenneth Arrow, Gerard Debreu, Lawrence Klein, Tjalling Koopmans, Harry Markowitz, Franco Modigliani, James Tobin, and other leaders in their fields. See Arrow et al. (1991).

At RAND in 1954 or 1955 I received a call from Professor James Tobin of Yale who explained that Tjalling Koopmans had decided that it was time for him to give up the administrative responsibilities of being head of the Cowles Commission. The search for a suitable new head of Cowles had the following result: The Cowles *Commission* was to move from Chicago to Yale, change its name from "Cowles Commission for Research in Economics at *Chicago*" to "The Cowles *Foundation* for Research in Economics at *Yale*." Professor Tobin would become its new head. Tobin invited me to spend the 1955/56 academic year at Yale writing a Cowles Foundation monograph on portfolio theory. A draft was finished during this period, subsequently reviewed by Tobin and Debreu with helpful suggestions from each, was revised and finally appeared as Markowitz (1959).

The 1955/56 academic year provided me the opportunity to focus on writing about portfolio theory and filling in some gaps in the theory as I saw them. In particular, Markowitz (1959) includes observations on how one or another model of covariance can be used in lieu of estimating individual correlation coefficients, as well as the definition of and computational procedures for using what I called "semivariance" (now sometimes referred to as "downside risk") in lieu of variance in risk-return analysis.

A principal preoccupation during this period was to reconcile portfolio theory with the theory of rational behavior under uncertainty as developed by von Neumann and Morgenstern (1944), Leonard J. Savage (1954), and others. Specifically, the problem was to reconcile the use of single-period mean-variance analysis by (or on behalf of) an investor who should maximize a many-period utility function. My answer lay in the observation that for many utility functions and for probability distributions of portfolio returns "like" those observed in fact, one can closely approximate expected value of the (Bellman 1957 "derived") utility function knowing only the mean and variance of the distribution. For details see Markowitz (1959) Part IV, Levy and Markowitz (1979), and Hlawitschka (1994); also Young and Trent (1969), Dexter et al. (1980), Pulley (1981, 1983), Kroll et al. (1984), Markowitz et al. (1994). Distinguish this view from the view that a Gaussian return distribution or a quadratic utility function is required for the use of mean-variance analysis.

After 1959. With the publication of Markowitz (1959) I had said what I had to say about portfolio theory, and published nothing further in the field for quite some time. My last substantial contribution to the early development of portfolio theory was to advise a young colleague at the RAND Corporation who was considering writing his dissertation (for UCLA) on portfolio theory. This led to his first publication: Sharpe (1963).

SPARSE MATRICES

Linear Programming at RAND. Shortly after I arrived at RAND I was approached by a small team of economists who wanted to apply linear programming to some RAND problem. They asked me to read Dantzig (1951) and supervise the programming and running of RAND's first simplex code. The programmer assigned to the project was Clifford Shaw, who later participated in the pioneering work on artificial intelligence with Herbert Simon and Alan Newell. The linear-programming problem posed by my economist colleagues was small by modern standards, with perhaps about 30 or 40 equations. If I recall correctly, the computer was called a "card-programmed calculator." I do recall Cliff Shaw often telling me that we did another iteration or two yesterday. He expressed optimism that some day we could do four iterations per day. I said that I would believe it when I saw it. We finally obtained an optimum solution.

George Dantzig arrived at RAND in 1952, about a year after I did. Linear programming and related technology made rapid advances during the 1950s, partly due to increased computer speeds and size and, in equal part at least, due to improved algorithms. See Dantzig (1963) for details.

Process Analysis and Sparse Matrices. Dantzig, Ford, Fulkerson, and Johnson were in RAND's math department; Alan Manne and I were in the economics department elsewhere in the RAND building in Santa Monica. Alan and I together with others, including Thomas Marschak at RAND and Tibor Fabian and Alan Rowe at UCLA, became interested in building industry-wide and multi-industry "process analysis" models of economic capability. Our objectives were similar to that of the Leontief (1951) "input-output" model, but our assumptions and methods were different. See Manne and Markowitz (1963) or Markowitz (1954).

Our models were constrained by the then-current size limitations on general LP models (about 200 equations max). It struck me that the constraint matrices for the models we had built, or were likely to build, consisted of mostly zero entries; that one could solve a modest-size system of equations by hand, without great difficulty, using Gaussian elimination if the matrix contained mostly zeros, and one carefully picked pivot elements so as to fill in as few as possible non-zeros when eliminating the row and column of the pivot; and perhaps this could be used to solve large linear programs with relatively few non-zero coefficients. I dubbed these "sparse matrices."

My solution to how to use sparsity in linear programming was based, in two ways, on the "product form of inverse" which Dantzig (1963) ascribes to a suggestion of Alex Orden. First, the product form illustrated that if one wanted to solve systems of equations with the same matrix and different right-hand sides, it was not necessary to actually have "the" inverse matrix. It was sufficient, and sometimes more convenient, to store the coefficients of a sequence of linear transformations which would transform any RHS to the solution of the equations. I viewed my procedure as developing a sparse such sequence of transformations which I called the "elimination form of the inverse." Second, in the simplex algorithm you do not solve several systems of equations with the same ("basis") matrix. Rather, you solve a system of equations using a given basis matrix A_1, then a system using its transpose A_1', then a system of equations with matrix A_2 that is identical to A_1 except for one altered column, etc. The purpose of the product form was to allow one to economically compute $(A_2^{-1}b)$ if one had A_1^{-1}. In particular, if one has a sparse constraint matrix one need not reinvert the basis matrix every iteration (or only solve specific equations without storing the inverse transformations), but can reinvert the basis now and then when the accumulated product form transformations grow too large.

I presented my sparse matrix methods in Markowitz (1957), including the results of an implementation programmed by William Orchard-Hayes. After that I lost track of what happened to sparse matrices until I visited IBM Research in Yorktown Heights, NY sometime in the early 1970s. Alan Hoffman told me that there had recently been a *second* conference there on sparse matrices, and that there was considerable work in the area (Willoughby 1969, Rose and Willoughby 1972; later, Duff 1981). I learned recently that the "Markowitz rule," for choosing pivots so as to keep the remaining matrix sparse after the Gaussian elimination step, is still used in at least one large production code for solving sparse matrix systems.

ENTITIES, ATTRIBUTES, SETS, AND EVENTS

SIMSCRIPT [I]. What we now refer to as SIMSCRIPT I (then referred to simply as SIMSCRIPT) was developed at the beginning of the 1960s at the RAND Corporation and made available without charge through the SHARE organization, see Markowitz et al. (1963). SIMSCRIPT was designed to facilitate the programming of "discrete event" simulation models, especially "asynchronous" discrete event simulators, as compared to continuous time or difference equation models.

The objective of SIMSCRIPT was to allow the simulation programmer to describe the world to be simulated, and relieve said programmer from implementation details insofar as we could. The SIMSCRIPT world view is as follows: As of an instant in time the system to be simulated has a *status* that changes at points in time called *events*. Status is described in terms of how many of various types of *entities* exist, what are the values of their *attributes*, and what entities belong to the *sets* which other entities own. Early 21st Century programming languages are likely to refer to Entities, Attributes, and Sets as Objects, Properties, and Collections (or Child-Parent relationships). Programming languages at the beginning of the 1960s spoke instead of variables and arrays.

The SIMSCRIPT [I] programmer described the entities, attributes, and sets of the system to be simulated on a Definition Form. In those days, the computer input was typically the punched card. The data written on the Definition Form, to be keypunched and placed in the SIMSCRIPT source program deck, included names of entity types; names of attributes, their data types, and precision information; the names of sets plus information as to what type of entity owns the set, what type belongs to it, and how the set is organized, e.g., FIFO, LIFO, or RANKED by one or more attributes of the members.

Changes in status were described in event routines written in the SIMSCRIPT programming language. The language included commands to CREATE and DESTROY entities, FILE entities into or REMOVE them from sets, FIND set members meeting specified tests, DO some action(s) FOR EACH member of sets, CAUSE or CANCEL subsequent event occurrences, etc., as well as perform arithmetic operations on attributes. We sought to make the commands English-like, "self-documenting." For example, to take specified actions on a set, like the jobs in the queue of some machine group, MG, one wrote

```
FOR EACH JOB IN QUEUE(MG)
            ⋮
REPEAT
```

where QUEUE(MG) is read "Queue of MG" just as $f(x)$ is read "f of x." In addition to the Definition Form and event routine language, SIMSCRIPT [I] had an Initialization Form for describing the initial status of the system and a Report Generator Form which provided a WYSIWYG output specification.

Prelude to SIMSCRIPT. SIMSCRIPT'S Entity, Attribute, Set, and Event view evolved gradually as the result of a great deal of simulation programming.

When I joined RAND in 1951, computer simulation was being used to evaluate bomber attrition given various offense-defense configurations, and other "war-game" situations. During the early to mid-1950s I sat in on discussions at UCLA of a group, including Alan J. Rowe and headed by Melvin Salveson, that sought to apply advanced analysis techniques to manufacturing planning. The consensus was that shop-wide and larger-scale manufacturing analysis was not amenable to analytic solution or optimization algorithms. Simulation analysis was required. Later in the 1950s, RAND created a Logistics Laboratory. Its first major project, LP1, was a man-machine simulation in which actual air force logistics officers played the role of air force logistics officers. The computer flew simulated missions, generated part failures and other maintenance requirements, and kept track of parts supplies and aircraft status. My job in LP1 was to coordinate the programming of the computer models.

Some time after LP1 was finished I got an offer I couldn't refuse from the General Electric Company, eventually moving to its Manufacturing Services at GE Headquarters in New York City. Alan Rowe was already there and had just supervised the programming of a large, detailed job shop simulator programmed in assembly language. Alan designed this simulator with a particular GE shop as an initial application, but with many input parameters whose specification were to tailor the model to other shops. But when Alan went to apply the model to a next GE shop it turned out not to be as general purpose as hoped.

Upon reviewing Alan's experience my own theory at the time was to reduce programming time and increase flexibility by building the next big shop simulator in FORTRAN II, constructing it as a system of reusable subroutines. I soon had the opportunity to test this theory. Together with a team of GE Transformer Department manufacturing engineers, and Mort Allen who programmed the model, I participated in the development of GEMS, the General Electric Manufacturing Simulator.

GEMS was well received within General Electric. Manufacturing Services conducted internal GE seminars on MSS (Manufacturing Systems Simulation) with GEMS as a principal topic, and we soon got a request to simulate another GE shop. In the process of building a simulator for this next GE shop it became apparent that GEMS was not as flexible as I had hoped. As to GEMS' reusable subroutines, the routines which proved most reusable performed basic actions like linking jobs into queues, or events into the calendar of coming events.

My next hypothesis was that such facilities could be placed at the disposal of the simulation programmer more conveniently as part of a simulation language rather than as subroutines. I didn't want to write this simulation language at General Electric, since it seemed (given my situation within GE at the time) that it might be deemed proprietary, for internal GE use only, and I wanted to see the ideas disseminated. I went into the job market seeking a new home for me and my nascent simulation language and ended up returning to RAND.

I search my memories but cannot recall when and under what circumstances—between the time of reviewing GEMS' experience and the time of designing SIMSCRIPT forms and commands—that the SIMSCRIPT mantra emerged, that "the world consists of entities, attributes and sets and changes with events."

SIMSCRIPT I.5. The SIMSCRIPT [I] language was implemented as a preprocessor into FORTRAN. Bernie Hausner programmed the preprocessor; Herb Karr wrote the programming manual (Markowitz et al. 1963). The three of us jointly designed the fine details of the language.

In 1963 Herb persuaded me to join him in forming California Analysis Centers, Inc., later Consolidated Analysis Centers, Inc., always CACI. Initially, we gave SIMSCRIPT [I] courses, and looked for contract work related to SIMSCRIPT applications. Separately, (that is, not through CACI), I consulted for RAND on SIMSCRIPT II development and other matters; Herb consulted for Douglas Aircraft.

CACI got a contract from IBM to make a version of SIMSCRIPT [I] for a new operating system. We used this as an opportunity to implement SIMSCRIPT as a compiler into assembly language rather than a preprocessor into FORTRAN, using an entity-attribute-set view of the compiling process which was concurrently being used at RAND in building the SIMSCRIPT II compiler. The new CACI version of SIMSCRIPT [I] also removed certain language restrictions that had been imposed on the original RAND SIMSCRIPT due to it being a preprocessor into FORTRAN, or due to our inexperience. We called the resulting product SIMSCRIPT I.5, since it was an advance over the original SIMSCRIPT but definitely not what was being developed as SIMSCRIPT II. One large CACI line-of-business for the next few years was the building of SIMSCRIPT I.5 compilers for various computers and operating systems.

SIMSCRIPT II. In 1968 or 1969 RAND released SIMSCRIPT II to SHARE and published its programming manual (Kiviat et al. 1968).[1] SIMSCRIPT II was not presented primarily as a simulation language, as was SIMSCRIPT [I], but as a general purpose language with a simulation programming capability. One conspicuous difference between SIMSCRIPT I and II is that the latter dispensed with the forms used by the former. For example, instead of the Definition Form, entities, attributes, and sets in SIMSCRIPT II are defined by statements such as EVERY MACHINE-GROUP HAS A NR_FREE_MACHINES AND OWNS A QUEUE. This was to avoid the logistics problems of supplying forms to users. If a new SIMSCRIPT were designed today, the Definition Form might be back—as part of a (now common) GUI (Graphical User Interface).

The original plan for SIMSCRIPT II was that it be documented and, to a certain extent, implemented in "seven levels." Kiviat and Villanueva summarize in their Preface to Kiviat et al. (1969) the functions of the first five levels as follows.

> Level 1: a simple teaching language ... Level 2: A language roughly comparable in power with FORTRAN ... Level 3: A language roughly comparable in power to ALGOL or PL/I ... Level 4: That part of SIMSCRIPT II that contains the entity-attribute-set features of SIMSCRIPT ··· Level 5: The simulation-oriented part of SIMSCRIPT II ...

Level 6 was to contain the SIMSCRIPT II database management facilities. The premise is that not only *simulated* worlds can be characterized by entities-attributes-sets and events, but so too the "real world" as represented by databases. Level 7 was intended to make the SIMSCRIPT II "language writing language" available to the sophisticated user to make special purpose extensions of the language. (See Markowitz 1979.)

The "SIMSCRIPT II" which RAND released to SHARE and documented in Kiviat et al. (1969) implemented Levels 1 through 5. Levels 1, 2, 3, 4, and 6 (5 omitted) were implemented within IBM under the name EAS-E. EAS-E is documented in Markowitz et al. (1984), Malhotra et al. (1983), and Pazel et al. (1983). We considered EAS-E to be very successful in terms of efficiency of execution and ease of programming as demonstrated by one real-world internal IBM application. However, I failed to convince IBM that it should release EAS-E as a product. IBM had recently converted from the hierarchical IMS database system to the relational System R with its SQL front end, and wasn't interested in launching a programming system based on a different data model at the same time.

Prelude to SIMSCRIPT II. SIMSCRIPT II improvements to SIMSCRIPT's simulation capabilities were primarily due to intensive use within RAND of SIMSCRIPT [I] for logistics system simulation. For example, SIMSCRIPT [I] had an ACCUMULATE statement which could be used instead of an assignment statement. Not only would ACCUMULATE assign the new value of the variable, but would also accumulate statistics, such as the min, max, and time-weighted mean and standard deviation of the updated variable. Inspection of the first real simulation programs written in SIMSCRIPT [I] showed a sizable fraction of the coding devoted to ACCUMULATE statements. It also made it clear that coding could be greatly reduced if the statistics to be accumulated were specified once, at Definition Time, rather than with each assignment.

SIMSCRIPT II was already on the drawing board before SIMSCRIPT [I]'s manual and preprocessor were completed. We knew we wanted SIMSCRIPT II to be rid of the SIMSCRIPT [I] forms, and to compile into assembly language rather than preprocess into FORTRAN. While we were at it, we wanted to remove some restrictions imposed on SIMSCRIPT [I] by implementation considerations, and restyle the language a bit to make it still more "self-documenting." SIMSCRIPT II is not an easy language for which to write a compiler. Consider, for example, how one can concatenate control phrases in SIMSCRIPT II, as in

```
FOR EACH MACHINE_GROUP IN SHOP
WITH NR_FREE_MACHINES
  (MACHINE_GROUP)> 0,
FOR EACH JOB IN QUEUE (MACHINE_GROUP)
WITH DUE_DATE (JOB) < TODAY
```

It is not required that FOR and WITH phrases be on separate lines as above. Line breaks and spacing on the page are inessential (except on the "form lines" following "PRINT *n* LINES THUS ...," which replaced SIMSCRIPT [I]'s Report Generator). FOR, WITH, WHILE, UNTIL, and UNLESS phrases can be combined in any meaningful manner, used to control single statements or blocks of statements demarked by DO ... LOOP statements; or they can be incorporated into FIND or COMPUTE statements. The latter computes MIN, MAX, SUM, MEAN, or STD_DEV at an instant of time, as distinguished from the ACCUMULATE statement which accumulates statistics across time.

I had heard that the compiler for some programming language was written in that language itself (perhaps JOVIAL in JOVIAL, but I am not sure I recall correctly). The idea of a SIMSCRIPT II compiler programmed in SIMSCRIPT II intrigued me. Of course, one would have to "bootstrap" a first version from SIMSCRIPT [I], but after that one could program more advanced versions in SIMSCRIPT II itself. While Bernie Hausner and Herb Karr finished the implementation and documentation of SIMSCRIPT [I], I made a first draft of the statements of SIMSCRIPT II and basic design of what the "SIMSCRIPT II-in-SIMSCRIPT II" compiler would be like. The latter includes an entity, attribute, and set description of the status of the compiler, and a language-writing-language wherein more complex commands, like FIND and COMPUTE, could be defined in terms of more basic commands. See Markowitz (1979) for details.

The first programmer that RAND assigned to the SIMSCRIPT II project struggled. After about a year Bernie Hausner returned to RAND from an extended leave and world travels. Bernie started the compiler over from scratch, programmed for a year or so until SIMSCRIPT II was capable of compiling SIMSCRIPT II. He then turned SIMSCRIPT II compiler development over to Richard Villanueva, assuring me that Richard was capable of completing the SIMSCRIPT II compiler, and he was. Bernie left RAND, joined the United Nations to supervise the building of a database system, and eventually became a U.N. diplomat. Meanwhile, back at RAND, Phil Kiviat agreed to write the SIMCRIPT II programming manual. At first the three of us—Kiviat, Villanueva and me—served as design team to specify remaining language details. Eventually, as my duties at CACI increasingly distracted me from the RAND SIMSCRIPT II development, the final language specifications as well as compiler development and the writing of Kiviat et al. were completed by Kiviat and Villanueva.

Eventually CACI established its own version, SIMSCRIPT II.5. Under the able guidance of Ed Russell this became the workhorse of the SIMSCRIPT simulation community for many years. All this was after CACI and I parted company.

March 15, 1968, and After. By the beginning of 1968 CACI had grown from Herb and me to a small but growing company planning to "go public." CACI's initial public offering did in fact take place during the second half of 1968. That was the good news. The bad news was that Herb and I had a major disagreement over the pricing of a new product, then a disagreement over how to settle disagreements. This was finally settled on March 15—the Ides of March—of 1968 when Herb Karr, with about 47% of CACI stock and Jim Berkson, vice president of finance, with about 5% of the stock, fired me with about 47% of the stock.

Currently, 33 years later, CACI continues to support SIMSCRIPT II.5. The February 2001 issue of *ORMS Today* (Swain 2001) includes a "Software Product Listing" for "Power Tools for Visualization and Decision Making." A line of the table for SIMSCRIPT II.5 includes the following entries. "Typical Applications of the Software: Building large, complex, high-fidelity, discrete event simulation models, with Interactive 2D graphics and built-in animation." The graphics and animation are new, i.e., were not part of SIMSCRIPT II, circa 1968. "Primary Markets for which the software is applied: Military theater-level simulations, telecommunications, factory simulations, hospital processes." "Price: PC Windows: $25,000. Unix: $35,000." "Vendor's Comments: Simscript II.5 Integrated development tools are based on famous language Simscript used worldwide for building portable, robust commercial quality simulation packages."

If it were up to me I would add database (i.e., Level 6) and distributed (e.g., internet) entities, and cut the price.

ENDNOTE

[1] Kiviat et al. has a 1968 copyright date and a Preface dated April 1969.

REFERENCES

Arrow, K. J., G. Debreu, E. Malinvaud, R. M. Solow. 1991. *Cowles Fiftieth Anniversary: Four Essays and an Index of Publications*. Yale University Printing Service, New Haven, CT.

Bellman, R. E. 1957. *Dynamic Programming*. Princeton University Press, Princeton, NJ.

Dantzig, G. B. 1951. Maximization of a linear function of variables subject to linear inequalities. T. C. Koopmans, ed. *Activity Analysis of Production and Allocation*. John Wiley & Sons, Inc., New York, 339–347.

——. 1963. *Linear Programming and Extensions*. Princeton University Press, Princeton, NJ.

Dexter, A. S., J. N. W. Yu, W. T. Ziemba. 1980. Portfolio selection in a lognormal market when the investor has a power utility function: Computational results. M. A. H. Dempster, ed. *Stochastic Programming*. Academic Press, New York, 507–523.

Duff, I. S. ed. 1981. *Sparse Matrices and Their Uses*. Academic Press, New York.

Graham, B., D. L. Dodd. 1951. *Security Analysis*, 3rd ed. McGraw-Hill Book Company, New York.

Hlawitschka, Walter. 1994. The empirical nature of Taylor-series approximations to expected utility. *Amer. Econom. Rev.* **84**(3) 713–719.

Kiviat, P. J., R. Villanueva, H. M. Markowitz. 1969. *The SIMSCRIPT II Programming Language*. Prentice Hall, Englewood Cliffs, NJ.

Koopmans, T. C. 1951. Analysis of production as an efficient combination of activities. T. C. Koopmans, ed. 1971. *Activity Analysis of Production and Allocation*, 7th ed. Yale University Press, New Haven, CT.

Kroll, Y., H. Levy, H. M. Markowitz. 1984. Mean variance versus direct utility maximization. *J. Finance* **39**(1) 47–61.

Kuhn, H. W., A. W. Tucker. 1951. Nonlinear programming. J. Neyman, ed. *Proc. Second Berkeley Sympos. on Math. Statist. Probab.* University of California Press, Berkeley, CA, 481–492.

Lemke, C. E. 1965. Bimatrix equilibrium points and mathematical programming. *Management Sci.* **11**(7) 681–689.

Leontief, W. 1951. *The Structure of American Economy, 1919–1931*. Oxford University Press, New York.

Levy, H., H. M. Markowitz. 1979. Approximating expected utility by a function of mean and variance. *Amer. Econ. Rev.* **69**(3) 308–317.

Malhotra, A., ——, D. P. Pazel. 1983. EAS-E; an integrated approach to application development. *ACM Trans. Database Systems* **8**(4) 515–542.

Manne, A. S., ——. 1963. *Studies in Process Analysis: Economy-wide Production Capabilities*. John Wiley and Sons, New York.

Markowitz, H. M. 1952. Portfolio selection. *J. Finance* **7**(1) 77–91.

——. 1954. Industry-wide, multi-industry and economy-wide process analysis. T. Barna, ed. *The Structural Interdependence of the Economy*. John Wiley and Sons, New York.

——. 1956. The optimization of a quadratic function subject to linear constraints. *Naval Res. Logist. Quart.* **3** 111–133.

——. 1957. The elimination form of the inverse and its application to linear programming. *Management Sci.* **3** 255–269.

——. 1959. *Portfolio Selection: Efficient Diversification of Investments*, 2nd ed. Basil Blackwell, Cambridge, MA.

——. 1979. SIMSCRIPT. J. Belzer, A. G. Holzman, A. Kent, eds. *Encyclopedia of Computer Science and Technology,* Vol. 13. Marcel Dekker, Inc., New York.

——. 1987. *Mean-Variance Analysis in Portfolio Choice and Capital Markets*. Basil Blackwell Ltd., Oxford, U.K.

——, Peter Todd. 2000. *Mean-Variance Analysis in Portfolio Choice and Capital Markets*. With Chapter 13 by G. Peter Todd. First published in 1987 by Basil Blackwell. Revised reissue by Frank Fabozzi and Associates, New Hope, PA.

——, B. Hausner, H. Karr. 1963. *SIMSCRIPT: A Simulation Programming Language*. Prentice Hall, Englewood Cliffs, NJ.

——, A. Malhotra, A. Pazel. 1984. The EAS-E application development system: Principles and language summary. *Comm. Assoc. Comput. Mach.* **27**(8) 785–799.

——, D. Reid, B. Tew. 1994. The value of a blank check. *J. Portfolio Management* **20**(4) 82–91.

Pazel, D. P., A. Malhotra, H. M. Markowitz. 1983. The system architecture of EAS-E: An integrated programming and data base language. *IBM Systems J.* **22**(3) 188–198.

Perold, A. 1984. Large-scale portfolio optimization. *Management Sci.* **30**(10) 1143–1160.

Pulley, L. M. 1981. A general mean-variance approximation to expected utility for short holding periods. *J. Financial Quant. Anal.* **16** 361–373.

——. 1983. Mean-Variance approximations to expected logarithmic utility. *Oper. Res.* **31**(4) 685–696.

Rose, D. J., R. A. Willoughby, eds. 1972. *Sparse Matrices and Their Applications. Proc. Sympos. IBM Res. Center*, September 9–10, 1971. Plenum Press, New York.

Savage, L. J. 1954. *The Foundations of Statistics*. Wiley, New York. 2nd ed., Dover, New York.

Sharpe, W. F. 1963. A simplified model for portfolio analysis. *Management Sci.* **IX**(2) 277–293.

Swain, J. J. 2001. Power tools for visualization and decision-making. *OR/MS Today* **28**(1) 52–63.

Von Neumann, J., O. Morgenstern. 1944. *Theory of Games and Economic Behavior*, 3rd ed. (1953), Princeton University Press, Princeton, NJ.

Wiesenberger, A., and Company. 1941. *Investment Companies*, Annual editions since 1941. New York.

Williams, J. B. 1938. *The Theory of Investment Value*. Harvard University Press, Cambridge, MA.

Willoughby, R. A., ed. 1969. *Proc. Sympos. Sparse Matrices Their Appl.* IBM Report RAI (No. 11707). Yorktown Heights, NY.

Young, W. E., R. H. Trent. 1969. Geometric mean approximation of individual security and portfolio performance. *J. Financial Quant. Anal.* **4**(June) 179–199.