

## Chapter 2

# Econometric Software: Characteristics, Users, and Developers

As the title indicates, this chapter takes as its major themes three aspects of econometric software, namely its characteristics, its users, and its developers. What it is, as a manifestation of its characteristics, is obviously quite relevant to this study as a whole, but so also are those who use it, how it is used, and by whom it has been created and developed. The users and developers of any type of software are clearly each formative influences, as may also be the particular way it is used, since collectively these circumstances shape and possibly explain its characteristics. Of course, to a degree, these are separable topics, or at the very least they are topics that can be considered progressively, beginning with the general characteristics of this software.

Following from the discussion in Chap. 1, there is a useful distinction to be made initially between the act of creation of software that is motivated by the goal of performing a specific task and the production of a finished program that is intended to be used for a more generally defined purpose and possibly by others. For instance, a given source code routine, or more often a set of routines, might be written specifically in order to calculate and display a set of parameter estimates. Alternatively, the goal might be to create a program to be used to build a “model,” or a class of models, or to perform some other composite, possibly quite complex extended task. Notice that what distinguishes these examples from each other is that the first defines what can be viewed as a *pure* computational problem, with success or failure to find the solution judged by the degree to which the specific calculations are accurately, efficiently, and even elegantly or perhaps quickly performed. What could also be included in this same qualitative assessment is whether, and to what degree, the results are informatively and even attractively displayed. In contrast, in the alternative, more general case, what is involved and needs to be evaluated is the potentially much more elaborate creation of the necessary computer code to perform an integrated series of tasks, not all of which are individually computational problems in the same pure sense. Furthermore, the performance of these tasks might also incorporate operations that only by an extreme stretch of imagination are likely to be classified as either economically or econometrically interesting – at least as the subject matter of these disciplines is usually defined. Nevertheless, the storage, retrieval and management of data, for example, as well as the development of the human interface of a program, are just as much a part of econometric software creation as is the programming of specific calculations. Moreover, whether all these aspects are

now individually considered to be econometrically relevant or not, the future development of economics and econometrics may nonetheless be affected by the degree to which each of these constituent software development problems is appropriately solved.

At first sight, the essential distinction between these two cases is a subtle one, at this stage even possibly obscure, making it necessary for clarity to recognize that the crux of the matter is the question of the extent to which the work performed should be seen by economists and econometricians to be within-discipline, not somebody else's concern. In the most general case, the econometric computational problem should be considered to comprise not only the process of performing specific calculations but in addition provision for the acquisition of an *appropriate* data set, the management and useful display of that data – possibly displaying both intermediate and final results – and maybe even the integration of a large number of computational tasks. In effect, in this general case, this computational problem can be defined as incorporating all the operational aspects of a given applied research project ranging from obtaining the data used from the original source or sources to the presentation of the published results, in whatever form these are presented – rather than just the implementation of specific formulae that might appear in an econometric text book or elsewhere in the literature.

However, although such an argument can be made, it can fall on deaf ears. Most often, economists have been focused in their research interests upon economic agents and their behavior, individually and collectively, and the econometrician upon such topics as the properties of particular parameter estimators, the circumstances of their use, and the evaluation of that use, frequently defined ideally. Consequently, econometric software development is an area of inquiry that is ordinarily interpreted to fall well outside the formal bounds of economic or econometric investigation, as a subject best left to the computer scientist or someone else, certainly as regards its detailed aspects. It is indicative that, with few exceptions, econometric textbooks and the more general econometrics literature ordinarily only refer to the existence of econometric software, without considering its specific characteristics. It is moreover telling that the various aspects of this software, even those most obviously “econometric” in nature, have seldom been considered and evaluated within this literature, except for relatively uninformative “software” reviews (McCullough & Vinod, 1999). Mainstream journals are not the usual locus of ongoing discussion of even the major developments most likely to affect applied economic research. Indeed, these journals normally reject independent submissions that focus too obviously on specific computational matters, even in those cases that these underlie and are critical to the evaluation of applied economics findings. The possible exception to this rule is numerical accuracy (McCullough & Vinod, 1999), but only recently and still in token form (Brooks, Burke, & Persaud, 2001; Bruno & De Bonis, 2004; McCullough, 1997, 1999, 2004; McCullough & Renfro, 1998; McCullough, Renfro, & Stokes, 2006; Stokes, 2004b, c, 2005). Economists ordinarily deal with computational issues at a full arm's length and depend upon others, often statisticians and the statistics literature, to intercede (Altman, Gill, & McDonald, 2004; Cordeiro, 2007; Hotelling, 1943; Longley, 1967; Simon and

James, 1988; Velleman & Welsch, 1976), notwithstanding that some related conceptual issues have been considered (Belsley, 1986, 1991; Belsley, & Kuh, 1986; Belsley, Kuh, & Welsch, 1980). Reflecting this situation, most economists are quite aware of John von Neumann's role in the development of game theory, but comparatively few know that he also wrote the first computer program and participated in the design of the first stored program computer (Knuth, 1970; Neumann, 1993/1945). Of these accomplishments, which ultimately will be judged to have had the greatest impact on economic progress is an interesting question.

In order to appreciate better the relevant issues, it is pertinent to consider in contrast another technological development, namely the first production of a printed book, using movable type, which occurred in 1455 or thereabouts. The invention of printing obviously has had a fundamental impact on the entire world of knowledge. More specifically, the transmission of information about economic and econometric thought and research historically rests upon the previous development of this technology, as does also the general diffusion of economic knowledge. Therefore it is not difficult to make a connection, as an enabling circumstance, between the development of the printed book and the development of economics or econometrics. Yet it is also immediately apparent that book production by or for economists or econometricians – either as regards particular technical production aspects or in terms of the publication or distribution of these objects – is only disciplinarily pertinent insofar as the specific focus of a particular investigation becomes their pricing or marketing, or perhaps the industrial organization of the book trade. Even then, it might still be difficult to identify those properties of the book as an object that would merit its isolated consideration as a uniquely characterized economic good. To make a convincing argument that the investigation of the technical aspects of book production appropriately falls within the discipline of economics or econometrics would thus appear to be rather thankless, notwithstanding that most economists spend much of their working lives participating in activities intimately associated with either the production or use of books in either hardcopy or digital form. In short, the example of the book illustrates that simple proximity is not the issue, nor are per se any historical developmental dependencies. On what grounds does econometric software present a better claim for attention?

## **Developmental Characteristics of Econometric Software**

The essential nature of econometric software can be considered prescriptively, following from a definition of econometrics and then proceeding to a determination of what characteristics this software does, should, or must have. An alternative approach, and the one that has been adopted here, is to define econometric software as consisting of that software intended to be used as econometric software; that is, software that is purposefully created by its designers and developers to be “econometric” in nature. This second definition of course accords with the

description in this volume's preface and in Chap. 1. However, if too literally interpreted, proceeding in this way risks a high degree of definitional circularity.

A possible solution to this potential circularity problem is to begin by considering the history of the development of this software, placing it in the context of the historical development of the electronic computer, as was done in Chap. 1, which considered also at least certain aspects of the particular economic and data processing environment from which both the computer and economic computation developed. These histories are not as separate as might be imagined. For instance, as the example of von Neumann illustrates, certain of the people closely associated with the design of the early computers and computer peripherals, if not obviously economists, are at least recognizable for their important contributions to the discipline or for the part they played in creating the context from which econometric computing came. In addition, it so happens that a number of easily identified economists and econometricians were among the earliest to start using the computer as a disciplinary research tool, use that has continued from then to the present day. This story also has several other interesting and relevant aspects that lead naturally into a consideration of the modern characteristics of econometric software.

### *The Early History of Econometric Computation*

The first use of computers by economists was a case of need meeting opportunity. It goes almost without saying that the 1930s and 1940s constitute a formative period for both economics and econometrics. During this period, in Britain, the United States, and elsewhere, the organized compilation of economic statistics by governments, trade organizations, and other such originating data sources began to be pursued in earnest, although at first much of this effort was actually expended by individuals. As described briefly in the Introduction, the time just after World War II more precisely marks the stage at which economic and social accounting first began to become fully institutionalized (Hicks, 1990; Kenessey, 1994; Kurabashi, 1994; Meade & Stone, 1941). These developments, although the culmination of an even longer process, were supported by a generally felt need to monitor and better understand economic phenomena, reflecting both the impact of world wars and the economic trials and tribulations of the 1920s and 1930s. In the 1930s, several statistically significant events of course occurred, among them the establishment of the Econometric Society and the publication of Keynes' *General Theory*. In addition, in 1936, not only was the *General Theory* published, but also Leontief's first work describing input-output relationships (Leontief, 1936). And, at about this time, Jan Tinbergen created the first macroeconometric models. Meanwhile, associated with the work of Ragnar Frisch, Trygve Haavelmo, Tjalling Koopmans, Richard Stone, and a number of others, there was also a quickening development of the methodology of econometrics and quantitative economics, which of course extended through the 1950s and beyond.

A revealing perspective on the circumstances of those times is provided by a comment incidentally made by Lawrence Klein in 1950 (Klein, 1950). Referring to the seminal Cowles Commission work and the methodological progress of the 1940s, he rather pessimistically observed (p. 12) that

An annoying problem that arises with the new methods is the laboriousness and complexity of computation. Very economical techniques of dealing with multiple correlation problems have been perfected, but they can no longer be used except in special cases . . . where the system is just identified. Unless we develop more economical computational methods or more efficient computing machines, the problems will remain beyond the reach of individual research workers.

This remark clearly demonstrates why economists were then ready to embrace the computer. Yet further evidence of this readiness is Leontief's purposeful participation in 1947 at one of the first gatherings of computer designers, the Symposium on Large-scale Digital Calculating Machinery, where he discussed certain aspects of his then-current work on interindustry relationships (Leontief, 1948). In his presentation, he described some of the challenging computational problems he saw that might be ameliorated by the use of such machines. Leontief was also during that time successively an active user of the IBM-supported electromechanical Mark I and II data processing and computational machines, in the process becoming the first economist to use a computer, in the particular sense of using an "automatic" device (Wilkes, 1956).

Other economists took similar advantage of the opportunities available to them. The first stored program computer to begin operation was the EDSAC (Electronic Delay Storage Automatic Calculator), independently built at the University of Cambridge by a team under the direction of Maurice Wilkes, yet in design a sibling of the EDVAC (Wilkes, 1956, 1985; Wilkes & Renwick, 1949; Wilkes, Wheeler, & Gill, 1951). The EDSAC was made available for academic research starting about 1951, the first electronic computer to be used in this way (Rosen, 1969; Wilkes, 1956; Wilkes & Renwick, 1949). Members of the recently formed Department of Applied Economics (DAE), several of whom had earlier been invited to Cambridge by its Director, Richard Stone, not only employed this machine (Aitchison & Brown, 1957; Farrell, 1957; Houthakker, 1951; Prais & Houthakker, 1955) but, in particular, Alan Brown, Hendrik Houthakker, and S. J. Prais (Brown, Houthakker, & Prais, 1953) appear to be the first to describe in print the process of using such a device as a disciplinary research tool. Lucy Slater, working with Michael Farrell at the DAE, created the first econometric software, consisting of regression and matrix manipulation programs for the EDSAC (Barker, Dada, & Peterson, 2004; Slater, 2004, 1962).

Computationally inclined economists elsewhere still operated electromechanical desktop calculating machines, including Lawrence Klein, Arthur Goldberger and their colleagues at the University of Michigan (Goldberger, 2004). However, in 1954, these econometricians were able to use the semi-electronic IBM Card Programmed Calculator (CPC), in addition to the 602A Calculating Punch, an electromechanical plugboard and card punch device, in order to estimate moments in preparation for the estimation of model parameters for the Klein-Goldberger

model (Klein & Goldberger, 1955; Sheldon & Tatum, 1973). A few years later, in 1958–1959, at what would become the Lawrence Livermore Laboratory, Frank and Irma Adelman were the first to solve an econometric model using a computer, an IBM 650 (Adelman, 2007; Adelman & Adelman, 1959; Knuth, 1986). Contemporaneously, possibly at the IBM Scientific Center in New York, Harry Eisenpress wrote the first program to perform Limited Information Maximum Likelihood estimation (Eisenpress, 1959), and a few years earlier, with Julius Shiskin in Washington, DC (Shiskin & Eisenpress, 1957), created the Census X-11 seasonal adjustment method using a UNIVAC (UNIVersal Automatic Computer), developmentally also a sibling to the EDVAC. This machine was built by another team under the direction of Eckert and Mauchly (Rosen, 1969). It was installed at the US Bureau of the Census in 1951 and was the first stored program computer to be sold commercially. Another UNIVAC was the machine used to predict the outcome of the presidential election in 1952.

### *The Takeoff Period of Econometric Software Development*

This description of early computer use by economists possibly appears to be a series of exhibits, selectively chosen, but actually this work effectively constitutes the complete early record, the salient exceptions being additional work by others at the DAE (Barker et al., 2004; Begg & Henry, 1998; Cramer, 2006; Prais & Aitchison 1954) and the beginning of Robert Summers' Monte Carlo study of estimator properties using an IBM 650 at Columbia University (Summers, 1965). The simple fact of the matter is that, throughout the 1950s, computers were scarce, difficult to gain access to, and expensive to use; an hour of machine time could cost literally triple the monthly salary of the economist using it (Adelman, 2007). As Bernard Galler has pointed out, "before 1955, any university that wished to establish a computing activity either had to build its own computer or have a special relationship with a manufacturer" (Galler, 1986). Consequently, only 47 universities had installed them by 1957 (Keenan, 1963). In addition, as mentioned in Chap. 1, the necessary programming infrastructure took time to build: recall that it was only in 1957 that Fortran, the first high level programming language, was developed (Backus, Beeber, Best, Goldbereg, Haibt, & Herrick, 1957); prior to that users needed to program in machine or assembly language. Hardware reliability was also a factor: only in 1959 did transistors begin to replace vacuum tubes (Rosen, 1969) and only in the 1960s did computers based upon this more reliable technology become available in the full sense of the word.

As a consequence, it was during the next two decades, starting in the early 1960s, as computers began to proliferate and programming languages and facilities became generally available, that economists more widely became users (Bodkin, 1999; Desai, 2007). Beginning then, there were a number of econometric firsts, including the implementation of increasingly computationally complex econometric techniques, among them Two and Three Stage Least Squares, Seeming Unrelated Regression Equations, and Full Information Maximum Likelihood (Renfro,



2004a, b, c, d). In addition, starting about 1964, economists created some of the earliest large scale computer data bases (large scale for that day), also developing the software to manage these (McCracken, 1966, 1967a, b). However, even in the mid 1960s, the progress made was neither uniform nor universal: the Wharton model, a direct descendent of the Klein-Goldberger model (Bodkin, Klein, & Marwah, 1991; Intriligator, 1978), was then still being solved using an electromechanical desktop calculator (Renfro, 2004a, b, c, d; Schink, 2004). It was only in the second half of that decade that economists at the University of Pennsylvania first used the electronic computer to solve large-scale macroeconometric models as nonlinear simultaneous equation systems, rather than as “linearized” reduced systems (Desai, 2007; Evans & Klein, 1967, 1968; Preston, 2006; Schink, 2004).

During the 1960s, expanding bootstrap-like on what had been learned, and making use of the ongoing technological developments, the level of sophistication progressively increased. Much of this work represented the efforts of individual economists, although often in the context of formal research projects (Duesenberry, Fromm, Klein, & Kuh, 1965, 1969; Evans & Klein, 1967, 1968; McCarthy, 1992; Preston, 2006). This process began with the creation of single purpose software to estimate parameters, manage data sets, and later solve macroeconometric models (Renfro, 2004a, b, c, d), but at the end of this decade, with the advent of time-sharing computers, economists were among the first to create and implement network-resident interactive software systems (Renfro, 1970, 2004a, b, c, d), a significant initial step in the development of the modern econometric computational environment. Beginning in the early 1970s, they made several concerted attempts to give focus to this research. At the just founded MIT Center for Computational Research in Economics and Management Science, under the direction of Edwin Kuh, economists began to devote considerable effort to the study of relevant computer algorithms (Berndt, Hall, Hall, & Hausman, 1974; Dennis, Gay, & Welsch, 1981a, b; Dennis & Welsch, 1978; Holland & Welsch, 1977; Kuh, 1972, 1974; Kuh & Neese, 1982; Kuh & Welsch, 1980) and closely related regression diagnostics (Belsley, 1974; Belsley, & Kuh, 1986; Belsley et al., 1980). Following such advances, during the 1970s economists then proceeded to develop several wide-area online telecommunications-linked economic data base, analysis and econometric modeling systems that by the end of that decade became used worldwide (Adams, 1981, 1986; Drud, 1983; Renfro, 1997a, b, 1980a). In other places, such as at the London School of Economics, the development of software played an integral part in the improvement of the methodology of specification search (Pesaran & Pesaran, 1987, 1997; Pesaran & Slater, 1980), with the software specifically conceived to be used as a tool to foster the so-called General-to-Specific, or LSE, method (Hendry, 2003b; Hendry & Doornik, 1999a, b; Hendry & Srba, 1980; Krolzig & Hendry, 2001; Mizon, 1984). As a consequence of initiatives of this type, including numerous smaller scale, even individual efforts, economists were then ready to adopt the emerging microcomputer at the beginning of the 1980s, by the middle of that decade starting to use it widely as the primary locus for analytical processing, including even the solution of econometric models of 600 and more equations (Renfro, 1996).

However, to provide the proper perspective for a more detailed evaluation of the implications of these advances, it is first necessary to place them in their appropriate historical context. The mention of microcomputers here is potentially misleading, given their ubiquity today, for although it is true that the first microcomputers became available to “hobbyists” as early as 1975, as a general proposition the 1970s were still very much the age of the bigger machines, for economists as well as most other people. Only later in this decade did economists begin to write software for the microcomputer (Renfro, 2004a, b, c, d). Furthermore, many of the econometric software initiatives just mentioned took almost 10 years to come to term, so that considering 1970s computer use in context it is important to recognize that the predominant computational characteristic was not only its mainframe focus but, as a matter of use, also a continuing selectiveness. The 1960s were a time during which economists more generally became computer users, as suggested earlier, but this was a change relative to the use in the 1950s. In 1970, only a small number of economists, or people at large, had directly begun to use the computer. In the case of economists, many were then graduate students, and of these only a proportion customarily spent each night and weekend in the keypunch room. Today, in contrast, computer use of course begins at kindergarten, or even before, and extends to any age pervasively, but such behavior began no earlier than the late 1980s.

Today there is also stress upon user accessibility. The human interface is much more evolved, in a way that permits the user to operate programs at a much higher level of abstraction. In contrast, programs in the early 1970s often, and in the later 1970s sometimes, still needed to be operated by placing numbers in fixed fields of punched cards, a given integer number from 1 to  $k$  indicating which among  $k$  options the user wished to select – or by the use of 0 the user might indicate omission. Sometimes data transformations needed to be made explicitly, the user coding these in Fortran or other high level language. One of the most significant differences between then and now is that in those earlier times, especially in the years prior to 1975, it was almost always requisite for a user to begin by writing some code, particularly in those instances that an application involved new techniques. So much was there a common need to program, and comparatively so much less computer use than today, that practically any econometric software written prior to about 1975 involves some aspect that can be declared a “first,” or at minimum viewed as involving some type of pioneering effort. There are certainly examples of off-the-shelf software used by (comparatively small groups of) economists during the period between 1960 and 1980 (Bracy et al., 1975; Brown, 1975; Goldberger & Hofer, 1962; Hendry & Srba, 1980; Kim, Chung, & Kim, 1979; McCracken, 1967a, b; Press, 1980; Slater, 1967, 1972); nevertheless, it was then usual that an intending hands-on user needed to learn to program, to at least some degree and often at a rather fundamental level. Until at least the middle 1970s, an applied economist would commonly either start with nothing more than a textbook, or, at best, be given a box of cards, or sometimes one or more paper tapes, onto which were punched the source code statements for a program – although by 1975, perhaps even earlier, it was not unusual for the “box of cards” to have been replaced by a machine-readable card image file.



However, in 1970, there were also certain incentives that led individual software developers to begin to focus on the human interface and abstract symbolic processing, as well as large scale data management, whereas others created particular algorithms or programs that since have been developed effectively as representative of what might almost be considered econometric “schools” of thought. Yet other economists and econometricians, as just indicated, created or modified particular programs for their personal use, certain of which ultimately became publicly available and widely used. Particular examples of programs of the first type, not all of which are still being developed, include DAMSEL, EPS, MODLER, TROLL, and XSIM, each of which were then associated with the creation, maintenance and use of often sizeable macroeconometric models (Renfro, 1980a). The use of such models characteristically imposes the need to create and maintain relatively large time series databases, involves the processing of symbolic data in the form of equations, and establishes a requirement for software that can be used by teams of people, hence the incentive to focus on the human interface. Examples of programs of the second type, individually interesting because of their internal algorithmic characteristics in the 1970s, include B34S, TSP, and Wysea, although TSP is also notable for the early development of its human interface. Programs of the third type, those that can be construed to be individually associated with a particular econometric “school,” include AutoBox and AUTOREG (represented today by its direct descendant PcGive). Finally, programs created originally in the 1970s for local, even personal use, but that have since been developed for public use, include AREMOS, CEF, FP, LIMDEP, MicroFit, Modeleasy+, RATS, REG-X, SHAZAM, SORITEC, and WinSolve. Some of these might also be regarded as being assignable to a “school.” These pigeonhole categories should all be regarded as being only tentative, but they nevertheless illustrate aspects of econometric software development before 1980.

During the 1970s, there were also larger forces propelling the development of econometric software. Overall, the economic spirit of that time was distinctly activist. At the end of the 1960s, in keeping with the particular Keynesian paradigm then prevailing, not only was there some feeling among economists that the economy was possibly precisely manageable (although possibly not to the degree this belief has been represented since), but – just as important – there was also a broader willingness on the part of government officials and corporate leaders, particularly in the United States, to believe in the capability of the economist to “fine tune” the economy. All this was to a degree the consequence of the fact that then the memory was still vivid of both the 1930s depression and the escape from it in the 1940s and 1950s. In 1945, it was popularly believed, an expectation shared by a number of economists, that an economic downturn was likely, that World War II possibly represented a temporary period of “full employment” that would give way to widespread unemployment when “the troops came home” (Klein, 1946; Orcutt, 1962; Woytinsky, 1947). However, 25 years later, at least in the case of the major industrialized countries, the recessions that had occurred had proved to be of short duration and represented minor fluctuations about a distinctly upward trend, particularly in comparison to before the war. One of the consequences, in the later

1960s, was substantial forthcoming corporate and government support for the creation of economic consulting firms such as Data Resources, Chase Econometric Forecasting Associates, and Wharton Econometric Forecasting Associates, each of which about 1970 began to create and support large scale, computer-resident economic data bases and econometric software systems (Renfro, 1980a, 2004a, b, c, d).

The development of econometric software during the period between 1960 and 1980 can therefore be seen as reflecting two distinct “motivating forces”: those internal to econometrics and those external, often deriving from economics as the parent discipline. Individual econometricians characteristically created software dedicated to parameter estimation using a variety of estimators; the academic imperative driving this type of innovation was of course the usual desire to present the new and different. For the major economic consulting and forecasting firms, the imperative was to provide billable services to clients; that is, to applied economists in governments, corporations, and other organizations. Software developed for the use of these clients was usually motivated by the goal to provide time-sharing software services in combination with access to substantial time series economic data bases, in many cases via telecommunications links (Adams & Ross, 1983; Mendelsohn, 1980; Renfro, 1980a). An important aspect of this software development was its emphasis on the creation of semi-natural language, symbolic command interfaces of the type that can be found even today as macro languages, intended to be easy-to-learn and easy-to-use (Adams, 1981; Drud, 1983; Kendrick & Meeraus, 1983; Meeraus, 1983; Renfro, 2004a, b, c, d). Another important aspect of this work, reflecting the widening range of users – many of whom might be less likely to call themselves econometricians than simply economists or even planners, statisticians, or something else – was the creation of more broadly-based facilities to support onscreen tables, graphs, and even maps, although not yet at today’s standards.

Both types of software development have proved to be important ultimately, each in its own way. Individual software development, for self-consumption, does not normally result in programs that can be used easily by other people, no matter how econometrically interesting the algorithms created. In contrast, programs developed intentionally for general use tend, by design, to offer not only “user friendly” interfaces, but also even extensive data display capabilities, which actually can be quite important in applied research, even if this type of software is open to criticism for “usually [lagging] some years behind the state-of-the-art technical econometric frontier” (Hendry, 1993, p. 314). The academic payoff, such as it is, tends to be much greater for software development that leads to the publication of information about new econometric technologies or embodies what are perceived to be theoretical advances, but this return represents a private, not necessarily a social benefit, beyond the creation of new knowledge that may or may not be of wide interest. In contrast, the greatest social benefit may derive from the implementation of new computational technologies that support the research of all economists and econometricians alike.

The particular historical details matter. The 1970s were years of substantial aspiration, yet also a time when the economist’s reach might be seen to exceed his grasp. Among the reasons this circumstance needs to be recognized is that

economists have been known to indulge in a form of poetic license. Specifically, in introductions to journal articles and other publications, or in the body of the piece, they have too often provided stylized, frequently not particularly well-founded statements of “fact,” intended as motivators for the argument presented. But even if not originally written to be definitive history, nor expected to be wholly believed, at least some of these statements have later tended to be cited repeatedly and in the end read as gospel, finally entering into the folk wisdom of the discipline. It is easy to demonstrate that the perspective provided by contemporaneous statements about technology, especially when read 10, 20, 30 or more years later, can be materially deceptive. For example, John Diebold, writing in 1962 (Diebold, 1962), presented an enthusiastic description of the capabilities of the language translation programs of that generation, asserting that they could be used to “scan a printed page, translate its contents from one language to another, make an abstract of the translations and store both text and abstract in ‘memory’ until they are called for by an information-retrieval network” (pp. 40–41). To anyone historically knowledgeable, this description clearly involves more than just a touch of hyperbole, as it would even if applied to modern language translation programs 45 years later.

In addition, it can be difficult even for those who were there to recall accurately all the relevant characteristics of the past when there has been substantial technological change in the meantime. Modern readers of an historical account may be even more inclined to assign present day capabilities to the past, making it difficult for them to separate historical fact and fiction. For instance, also in 1962, Daniel Suits wrote about macroeconomic models (Suits, 1963), asserting that “in the days before large digital computers, individual relationships generally had to be kept simple, and the number of equations that could be used in a system was rather small. Today [that is, 1962], with the aid of high speed electronic computers, we can use models of indefinite size, limited only by the available data” (p. 7). For proper perspective, it needs to be appreciated that this assessment precedes by several years the first successful computer-based solution of any econometric model, the single special-case exception being the Adelman work mentioned earlier. Furthermore, from 1962 to 1965, *every working macroeconomic model* (of the few operating) was solved using electromechanical desktop calculating machines, not computers. Even including the Adelman simulations, before 1965 model solutions were invariably enabled only by linearization of the model and then by first solving out all but a few variables. As indicated earlier, modern methods of solution only began to be implemented in 1967 (Desai, 2007; Preston, 2006; Schink, 2004); even then, in certain cases, desktop calculating machines continued to be used. Suits’ assessment was written before *any* software systems had been developed that had the capability to manage *to any significant degree* computer-resident economic data bases or to estimate, particularly at the scale suggested, the parameters of individual relationships or to handle any of the other computer-related tasks necessary to support the use of even small models, much less those of “indefinite size.” If unrecognized, past technological mis-assessments can cause hard-won successes to be undervalued and the causes of past failures to be misjudged.

### *The Adoption of the Microcomputer*

The computers ordinarily used by economists prior to the middle 1980s can be classified as mainframes or minicomputers, although by then supercomputers had also appeared, after being introduced in the 1960s, even if seldom used by economists. Mainframes were of course intended (and priced) for “enterprise” use, whereas minicomputers were meant to be used in departments and other, generally small, sub-classifications of organizations. Supercomputers, typically a category of fast, vector processor machines in the 1970s, had the interesting characteristic that they generally used mainframes as auxiliary processors for the input of data and output of results. However, irrespective of such classification, the fundamental characteristic of econometric computing before the later 1980s was not only that the computers used by economists were ordinarily organizationally owned, rather than personally, but also that they were shared. Mainframe sharing at this time meant that, however fast the machine’s operation when used by a single person, it could be quite slow for the average (multi)user. It might be slow because it operated in batch mode, with programs prioritized and queued as they were read in and, on output, the hardcopy results sorted and distributed by hand by its one or more operators. In many contexts, output in those days was essentially hardcopy, notwithstanding that it was possible to create a permanent or semi-permanent machine-readable record. The result could be turnaround times of an hour or more, or even 24 hours for “large” jobs, those requiring memory in excess of 512 KB or, sometimes, as little as 256 KB. But even when machines were used in “time-sharing” mode, the fact that individual users’ “jobs” were still almost always prioritized and then queued, and might require a human operator to mount tapes and removable hard disk drives, could mean that several minutes or more might pass between the entry of a single command and the computer’s response. The first personal computers were themselves slow, compared either to mainframes or minicomputers (or personal computers today), but they were single user and self-operated. These characteristics caused them to be time competitive with mainframes, and sometimes even supercomputers, as early as 1981 or 1982 (Fried, 1984, p. 197).

The adoption problem the microcomputer initially posed for the econometrician was the lack of software, which always occurs when the hardware characteristics are radically changed because of the introduction of an entirely new Central Processing Unit (CPU). In addition, the architecture of this new class of computer at first also represented a significant step back in capabilities: maximum memory size on the order of 64 KB, rising to 640 KB only more than a year later, and small, slow diskette drives for permanent storage rather than hard disks, with hard disks initially unavailable and then reaching the size of 20 MB, as a common characteristic, only in 1984 – not to mention CPU operating speeds of 6–8 MHz or less before 1986 (Byte, 1984, 1986). Furthermore, it took several years before microcomputer software provided the capabilities of mainframe software, reflecting that writing software takes time, but also that the language compilers and linkers available for the microcomputer were at first “bug” ridden, idiosyncratic, and originally

designed for small, memory-undemanding programs. In at least one case, in 1982, it was necessary to “patch” an existing linker in order to make it possible to convert an econometric software package from the mainframe to the PC.

Nevertheless, by the autumn of 1983, the first econometric software package capable of estimating and solving (small) econometric models was available for the IBM Personal Computer and compatibles. In September 1984, a microcomputer based economic forecast service was introduced at the annual meeting of the National Association of Business Economists, combining a 250 + equation Wharton Quarterly Econometric Model of the United States with the MODLER software (Renfro, 1996). The solutions for a 12 quarter forecast horizon took less than 4 min. This software had the same capabilities as its mainframe version (Drud, 1983); in particular, it could be used to create, maintain and solve econometric models of as many as 1,000 equations. By the end of 1985, other packages available for the “PC” included AREMOS, AutoBox, Gauss, PcGive, RATS, SHAZAM, SORITEC and Stata, as well as limited versions of both SAS and SPSS. Even earlier, at the beginning of the 1980s, more limited packages had been implemented on both a Tandy machine and the Apple II (Renfro, 2004a, b, c, d), including a program called “Tiny TROLL,” created by Mitch Kapor at MIT, parts of which were then incorporated into the VisiCalc spreadsheet package and subsequently also influenced aspects of the development of Lotus 1-2-3, and later other packages, such as Excel.

Many of these individual efforts continue to have a modern day relevance, but to explain the subsequent evolution of this software during the present microcomputer age, it is possible to trace the broad outlines of the computational developments of the past 20–30 years. The computational shift during the 1980s, from the creation of software and systems on large institutionally based machines to the use of the personal computer as the locus of such work, can be viewed as responsible for the range of econometric software that exists today. The personal computer, because of its affordability, ultimate wide distribution, and steadily increasing capabilities, not only provided an important context but also became the basis of a progressively more extensive market. The 1960s may have been the decade that economists first began to learn to use the computer, but it was the 1980s and subsequently that computer use became widespread in a pervasive sense. The comparative degree of market extensivity is even more apparent today, given the ubiquity of the notebook, or laptop, computer, and otherwise the sheer number of personal computers now commonly found in offices and homes, not to mention such things as the recently accelerating convergence of television and computer technologies. Of course, the development of the Internet as an effective successor to the more local, mainframe-based wide area networks of the 1970s has obviously had a significant impact, particularly since the middle 1990s, especially on the distribution of economic data and information.

Consequently, although it is possible to talk in terms of nearly 60 years of evolution, the impetus for the development of today’s number and variety of econometric software packages is decidedly more recent. Their present characteristics are the direct result of a combination of relatively modern circumstances, among them being the introduction of the microcomputer in the 1970s and 1980s, the essentially simultaneous expansive development of econometric techniques since the 1960s

(Gilbert & Qin, 2006), and most recently the increasingly common adoption of a graphical interface, often while preserving macro language capabilities, in conjunction with the progressively more widespread use of the Internet since the 1990s. Among the effects, the broadening and deepening of econometrics – and, more generally, quantitative economics – especially during the past 30 years, has had a significant impact on the range of the present day properties of these programs, resulting in considerable diversity. For example, functionally classified, today they can be placed in categories that include basic regression, advanced estimation, and econometric modeling languages. Considered in terms of both functionality and interface, they can be classified as ranging from those defined by specific-selection, menu-oriented econometric features to algebraic quasi-natural language econometric modeling and programming languages that provide also the capability for an individual user to create new techniques (Renfro, 2004a, b, c, d, p. 59ff).

Substantive changes in hardware have also occurred during the past 20 years. As indicated, the desktop Personal Computer in 1987 operated at 6, 8, or 10 MHz; in contrast, many modern notebooks operate at or near 2 Ghz or better. The 1985 microcomputer ordinarily contained at most 640 KB of easily accessible memory; today's variety commonly contains as much as 1 gigabyte or more. Furthermore, the microcomputer has progressed to the point of incorporating (in a single chip package) even two to four processing units (with the prospect of eight or more in the foreseeable future), as well as having other characteristics that make it more and more difficult to conceptually distinguish between the capabilities and types of large and small machines in a meaningful way that does not involve mind-numbing detail. What is certainly true is that the microcomputer found either on the desktop or an airline tray table is now the locus of the vast proportion of all the empirical analysis that is done by economists. In almost every sense, the composite history of the electronic stored program computer is now present in the modern personal machine.

## **The Characteristics of Econometric Software**

To this point the focus has been upon the developmental characteristics of econometric software during the past nearly 60 years. An inference that might be drawn is that, on the one hand, there are historical examples and, on the other, modern examples. It also might be thought that the historical examples are only of historical interest. To a degree, this characterization is reasonable: a number of econometric software programs were created, used for a period of time, often years, and then dispensed with. However, to a significant degree it is misleading. None of the programs created in the 1950s are still employed today, but certain of those from the 1960s continue to be, although in modern form. In particular, AutoBox, B34S, Microfit, MODLER, Mosaic, PcGive, TSP and Wysea all have their origins then and at least some still incorporate a certain amount of original source code. Furthermore, the majority of these programs continue to be maintained by their original principal developers. Others,



including AREMOS, FP, LIMDEP, Modeleasy+, RATS, SHAZAM, and SORITEC began life on mainframes in the 1970s, as did also SAS, SPSS, and other well-known statistical software packages; in some cases, these too continue to be maintained by their original developers. All were converted to microcomputers, in most cases beginning at various times during the period 1980–85. In contrast, REG-X began to be developed on a Tandy microcomputer in 1979, was moved to a mini-computer and then to the PC in the 1980s. Others, including EViews (as MicroTSP), Gauss, and Stata, began to be developed on the microcomputer in the 1980s, joined by Betahat, EasyReg, Ox, and the present day incarnation of TROLL in the 1990s. Of all the recognized existing programs, only gretl began to be developed in the present century, albeit on the basis of “inherited” code, even if there are also certain Gauss and Ox-based special applications that have been created during the past few years. The packages just identified include those that have been surveyed and are evaluated in this monograph.

Inasmuch as the origins of most date from before 1980, their history and that of the electronic computer are intertwined. Econometric software spans the development cycles of hardware change from earliest times. For instance, in addition to the early use of the computer described in the first part of this chapter, the first use of the computer by economists at the University of Pennsylvania apparently involved some later use of the UNIVAC during the early 1960s, the immediate design successor to the EDVAC (Desai, 2007; Preston, 2006), although this is almost incidental. However, the connections to the second generation are quite meaningful. At least three of the existing packages began to be developed on second-generation computers, and several more on the third. The distinguishing hardware characteristic of the second generation was the introduction of the transistor, which occurred first in 1959 with the IBM 7090/94 (Rosen, 1969). Another machine, the IBM 7040, was effectively an IBM 7090 “lite.” The IBM 1130 and 1620, used in several cases by economists, were second generation, small mainframes principally designed for scientific use. The CDC 6400, used in at least one case, can be described as a second generation mainframe, although it is architecturally compatible with the earlier CDC 6600, designed by Seymour Cray, which is generally regarded as the first supercomputer. Interactive, local area econometric computing began in 1970 at the Brookings Institution on a Digital Equipment PDP-10 (Renfro, 1970), another of which was later used by Bill Gates and Paul Allen ([www.pdpplanet.com](http://www.pdpplanet.com)). The IBM 360 was a third generation machine and was used by econometric software developers, as were also its successors the IBM 370 and the 3090. Other econometric software developers, especially those in the United Kingdom, if they did not actually cut their teeth on the first EDSAC, can nevertheless date their earliest work to the use of the Atlas in the 1960s, particularly the machines at the Universities of Cambridge and London, or even the EDSAC 2 or Titan (Slater & Barker, 1967). More recently, econometric software has involved the use of Apple, Tandy, the Victor 9000, the RS/6000, several Sun machines, and multiple generations of the IBM PCs and compatibles. The inference to be drawn is that econometric software enjoys a long and rich hardware patrimony, one only partially described here.

However, until recently, this history has been part of the econometric deep background. Only certain individual developers have ventured into print to any significant degree (Belsley, 1974; Eisner, 1972; Eisner & Pindyck, 1973; Hendry & Doornik, 1999a, b; Hendry & Srba, 1980; McCracken, 1967a, b; McCracken & May, 2004; Renfro, 1981, 1996, 1997a, b, 2004a, b, c, d; Slater, 1962; Stokes, 2004b, c; White, 1978). Furthermore, although the user guides and reference manuals commonly provided with individual programs often do give some information about their history, these accounts tend to be presented selectively, ordinarily without technical details. The most readily available, collective description of the existing econometric software packages, albeit somewhat limited, is found in the compendium published in 2004 (Renfro, 2004a, b, c, d). This collection comprises edited accounts by each of the current principal developers of each of the existing packages, although there are certain exceptions to this rule. The exceptions occur mainly in the case of historically significant programs that are today no longer maintained. Other, more selective, descriptions of particular econometric software packages, available in 1983 and earlier, can be found in an article of that date by Arne Drud (Drud, 1983), articles in a special issue of the *Journal of Economic Dynamics and Control* (Kendrick & Meeraus, 1983), and in minimally descriptive compilations of statistical software by Ivor Francis and others (Francis, 1981).

It is said that the past is a foreign country, but if the detailed, step-by-step record is now difficult to recover entirely, it is possible to determine the salient characteristics of these packages during modern times on the basis of an earlier interactive survey made in 2003. This survey was taken in conjunction with the publication of a special volume on econometric computing, published in 2004 both as volume 29 of the *Journal of Economic and Social Measurement* and a separate book (Renfro, 2004a, b, c, d). A number of the more general operational characteristics of the individual packages are documented there in the form of summary tables (Renfro, 2004a, b, c, d). In addition, the compendium just referred to (Renfro, 2004a, b, c, d) is included. It is interesting that the transition from desktop calculators to the electronic computer that began to take place in the early 1960s originally occurred in the form of a modal transfer: calculations previously made with the calculator began to be made instead using the computer, but initially without a significant change in mindset (Desai, 2007; Goldberger, 2004; Slater, 1962). After that first step, came the process of incorporating into this use both more comprehensive data management and more than particular parameter estimation methods. As mentioned earlier, the first recognizable econometric software commonly took the form of separate, single purpose programs classifiable individually as data management, data transformation, and regression programs, the latter in their original form not always easily distinguished from “statistical” programs of that day. To the degree that evident differences existed in the middle 1960s, the most obvious characteristic of econometric software was less of a tendency to include stepwise regression and more to include simultaneous equation techniques, such as Limited Information Maximum Likelihood and Two Stage Least Squares. It was only in the late 1960s, and even then only occasionally, that the programs became more than rudimentary in operating style and econometricians even began to think about something as conceptually sophisticated as software “design.”

In contrast, during the past 25 years, reflecting the impact of personal computers, econometric software packages have become clearly categorically distinguishable, both from other types of software and from each other. Among themselves, as a general property, individual programs have become functionally more self-contained, combining parameter estimation capabilities with data transformation facilities and at least a minimal degree of more generalized data management and display capabilities, a number of packages increasingly integrating as well such capabilities as nonlinear multi-equation model solution facilities. Since 1995, there has also been a pervasive tendency to adopt the prevailing standards of the so-called “graphical user interface,” associated with both Microsoft Windows and the Apple operating systems, although just as noticeable it has also been common for econometric software to continue to offer command line control, usually in the form of a scripting or macro capability. Most programs are today able to operate by manipulating econometric objects using a keyword-based command language, even if many operate primarily using menus and icons. It is also common to permit users to collect command elements into a text file, as a macro. The motivation is the repetitive nature of many of the operations performed during research; for example, requiring the ability to make data transformations repeatedly as new observations are acquired, or to rerun regressions. The ability to recycle commands, in order to perform previously executed tasks easily and repeatedly, is obviously a desirable trait.

The way in which the various specific econometric techniques came to be embedded in software during the past 50 years can also be outlined and usefully classified. Certain of these developments represent a widening, or broadening, in the number of econometric techniques, tests, and other operations implemented in software. Others represent a capital deepening process, in the sense of more sophisticated implementations that, in some cases, take the form of more complete algorithms that subsume the capability to perform any of a multiplicity of more elementary operations, including two or more econometric techniques in combination. In other cases, this deepening involves combining in the same program a sequence of operations that are mutually integrated, such as permitting parameters to be estimated as a first stage operation, followed by the very nearly automatic creation of model equations, and then linking these equations, as a next stage, finally causing the creation of a functionally complete model capable of being solved (Renfro, 2004a, b, c, d). Such broadening and deepening can be considered to be algorithmic in nature, although as also involving stylistic elements.

However, another aspect of this software development took the form of the creation of progressively more sophisticated interfaces, as discussed earlier. One of these is the human interface, the means by which the program user both controls the operations performed and either perceives or comprehends the results, which may or may not be the same thing. As mentioned before, in the 1960s, sometimes even in the 1970s, program control was effected by choices made using numbers located in fixed fields on punched cards or paper tape. This type of control has long since been replaced by the use of the WIMP graphical interface (Windows, Icons, Menus, and Pointing methods) and even earlier by the use of free-form, if still stylized command languages. The results generated may, in turn, be displayed in tabular form, or

as graphs, or as other perceivable objects, such as an equation or a list of equations. Comprehension, as opposed to simple perception of the program's output, obviously can be aided by interface design, even if there has been considerably less attention paid by econometric software developers to this aspect of the human interface than to enabling simple perception.

Another interface type is the machine interface, the way in which a given computer either receives input from or sends output to one or more other machines. The idea of facilitating and then generalizing this interface, including its hardware aspects, so as to permit computers to intercommunicate effectively began to be implemented at the beginning of the 1970s, when it became progressively more desirable not only to connect individual users to machines remotely from a dumb terminal via a telecommunications link, either dial-up or dedicated, but also one computer directly to another. Peer to peer machine linkages were initially difficult to achieve, for computers in those days were originally designed to operate singly, not as either intelligent or co-equal correspondents. Connections then generally required some type of master-slave protocol. More recently, the machine interface has of course often taken the form either of a Local Area Network (LAN) or a Wide Area Network (WAN) connection, the latter including both the Internet and other machine-to-machine linkages. For econometric software developers, these were initially separated innovations, for ordinarily these developers were not involved in the establishment of machine interconnection protocols, as this is an operating system task. However, once these connections began to be possible, remote data retrieval and data base management, among other facilities, began to become important as ideas and in practice (Anderson, 2006; Anderson, Greene, McCullough, & Vinod, 2007; Harrison & Renfro, 2004; Renfro, 1980a, 1997a, b; Ritter, 2000), even if today it is still usual for econometric software packages to be designed simply to read in data from some type of text file or an Excel or some other spreadsheet file, rather than to query a relational or other remote data base system using SQL or other procedural language.

### *Aspects of the Evolution of Software Features*

Mary Morgan (Morgan, 1990) and Qin Duo (Qin, 1993) have each described the process of the development of econometric theory and the way in which the ideas of Frisch, Haavelmo, and Koopmans, among others, and the work of Tinbergen, Klein, Goldberger and others during the early days of macroeconomic model building combined to establish both econometric practice and its received theoretical support at the beginning of the 1960s. Qin's assertion (p. 65) that "estimation can be seen as the genesis of econometrics, since finding relationships has always been the central motive and fulfilment of applied modeling activities" expresses well what can be regarded as a motivating thought behind the beginning efforts to more generally employ the electronic computer in the first few years of the 1960s. However, the operative philosophical position of those years was often that expressed in

1958 by Haavelmo (1958, p. 351), that “the most direct and perhaps most important purpose of econometrics has been the measurement of economic parameters that are only loosely specified in general economic theory.” Of course, this measurement often took place without always sufficiently taking into account his clearly stated qualification (p. 352) that the quantification of economic phenomena had in the preceding 25 years appropriately come to be interpreted to extend “not only to the measurement of parameters in would be ‘correct’ models, but to the field of testing, more generally, the acceptability of the form of a model, whether it has the relevant variables, whether it should be linear, and many other similar problems.” The methodology debates at the end of the 1970s and into the 1980s stand as testimony to the continued lack of testing as a practice, which, as will be discussed in the next chapter, at least in part possibly reflected the slowness with which facilitating statistical tests became embodied in the software.

In the early 1960s, the electronic computer, as it became progressively more commonly available, represented to economists the potential to perform computations not feasible previously. Eisenpress’s creation in 1959 of a program that implemented limited information maximum likelihood was followed in 1962–63 by the efforts of Zellner and Stroud to implement the Two and Three Stage Least Squares (Zellner, Stroud, & Chau, 1963a, b) and Seemingly Unrelated Regression Equations (Zellner, 1963a, b) techniques. This work by Zellner and Stroud marks the first time that particular estimation techniques were contemporaneously introduced in the literature (Zellner, 1962; Zellner & Theil, 1962) and implemented in software that could be used by others. A short time after that, in 1963–64, Mike Wickens programmed Full Information Maximum Likelihood, based upon a later-published formulation by James Durbin (Durbin, 1988) that, among other things, utilized Newton-Raphson convergence and demonstrated that the second iteration of the process generated Three Stage Least Squares estimates. Elsewhere, during this time, other econometricians also implemented estimation techniques in software; much of this work took place in Canada, New Zealand, the United Kingdom, and the United States (Bodkin et al., 1991; Klein, 1960). In most cases, these efforts can be seen to be motivated by the desire to make these calculations specifically for the sake of it. Other efforts in the middle to late 1960s – including follow on work in New Zealand (Bergstrom, 1967a, b; Phillips & Hall, 2004), as well as the program development that took place in Washington, DC at the Brookings Institution (Duesenberry et al., 1965, 1969; McCarthy, 1992), and that at the Wharton School of the University of Pennsylvania (Evans, 1969; Evans & Klein, 1967, 1968; Preston, 2006; Schink, 2004) – represented much more the need to support the estimation, construction, and use of macroeconometric models. However, as this was the take-off period of econometric software development, being the first dispersed attempt to create a software infrastructure, in almost all cases the initial effect was broadening, rather than deepening, as more and more estimation and even model solution techniques became embodied in software.

A broadening also took place in the 1970s that in many cases and in similar ways at first represented the efforts of individual econometricians, yet has since resulted in the general availability of packages such as AutoBox, B34S, BRAP,

FP, LIMDEP, Microfit, PcGive, RATS, and SHAZAM. Recall that these programs appear to have originated either as individual reactions to the local unavailability, or simply the general absence, of appropriate software or else as solutions to one or more specific, perceived econometric problems, or, indeed, the combination of these circumstances. Sometimes, as in the case of MicroFit and PcGive especially, this software development increasingly over the years included the incorporation of misspecification tests and other evaluative features. But whatever its exact form, most of this broadening, beginning then and extending to the present day, constituted the addition of econometric techniques. However, these efforts did not simply represent an increase in the number of techniques to be applied in a given, possibly macroeconomic time series context, but, in certain cases, the development of software to be used instead in a cross-section or panel data, often microeconomic environment. The greater availability of survey data, both cross-section and panel, as well as econometricians' advocacy of Bayesian, Time Series Analysis, and other specific methodologies provided much of the initial broadening stimulus in the 1970s. In the 1980s and 1990s, the market possibilities provided by the microcomputer and, in later years, the Internet, added extra stimulus. However, some of these packages, even in their early days, also supported the applied research of economics departments and groups of economists at such diverse places as Auckland, the Brookings Institution, Chicago, Cambridge, Harvard, the London School of Economics, Minnesota, MIT, Pennsylvania, Princeton, and Wisconsin, so were not just being developed in isolation for their developers' personal research use.

The phenomenon of software deepening is both most evident and easiest to describe in the case of programs developed for research teams associated with large-scale econometric model projects. The need to manipulate and display substantial quantities of data in conjunction with the creation and use of such models, starting in the middle 1960s, led increasingly during the 1970s to the creation of large scale economic data base management systems, both separately and as sub-components of such packages as DAMSEL, EPS, MODLER, Mosaic, and XSIM (Renfro, 1997a, b). From the 1960s to the later 1980s, data series often needed to be acquired in hard copy form and then keypunched. The associated expense obviously provided an incentive to develop ways to move the data, once in machine-readable form, from one context to another with a minimum of effort, as well as to manipulate the observations easily. Models containing 300 or more equations only became possible because of the computer hardware and software advances that began in the 1960s, although at first models of this size certainly strained the existing computational capabilities. Even in the early 1970s, to create a 200 equation model was commonly held to require a year's effort on the part of a team of 10–12 people (McCracken & Sonnen, 1972). In 1987, in contrast, one person working alone could estimate, construct, and successively solve a 300 equation model in a single week (Cooper, 1987; Renfro, 2004a, b, c, d).

The objects that are associated with macroeconometric models containing hundreds or even thousands of equations obviously include data series, which explains the development of data base management capabilities. However, somewhat less immediately obvious, they also include equations, multiple tables, graphical displays,



macros used repeatedly to make transformations and updates, and other such items that also need to be managed effectively. These objects collectively constitute a significant data management problem that involves not simply classification and organization, but also a general problem of information management that includes the need to be able to search effectively. In addition, from the beginning there was a requirement to incorporate labor saving features; for example, the manual coding of individual model equations itself was time consuming, but in addition likely to result in transcription errors. Otherwise, in common with other types of software, deepening in this context also took the form of the creation of program components capable of performing a variety of selected transformative operations on a particular data input stream (Hendry, 1976). As indicated earlier, this intensification process can be considered both as an internal program phenomenon, as just briefly described, or else in connection with the development of human command interfaces that make possible the more sophisticated control of a program's operation.

### *The Development of the Human Interface*

One of the evolutionary characteristics of econometric software – as discussed briefly earlier, and in greater detail elsewhere (Renfro, 2004a, b, c, d) – was the early development of explicit econometric modeling languages, which began in the late 1960s. The use here of the term “language” refers to the command structure as a human interface, which permits the user of this type of software to describe to the software the operations to be performed using an algebraic syntax and vocabulary, together with keywords and variable names; for example resulting in transformation commands (possibly simultaneously taking the form of identities) such as:

$$Y = C + I + G + (X - M)$$

The variable names (Y, C, I, G, X, M) not only have an obvious mnemonic aspect, but as command elements each constitutes also a symbolic reference to a stored vector of observations. The use of the program's command language therefore not only directly invokes the retrieval of observations from an organized data base, and perhaps subsequently the storage of results there, but also defines and causes calculations and other operations to be performed that can be associated with the construction, maintenance, and use of an econometric model, a model that might contain even hundreds or a thousand or more equations. However, once created, such a program can also be used more prosaically to make simple data transformations, as shown above, as well as to perform regressions, execute a variety of analytical tasks, display tables, graphs, and the like, all in a relatively user friendly way. Consequently, as previously described, the development of econometric modeling languages in the 1970s was often associated with formation of economic consulting and forecasting firms, which then made available to a wider public both software services and economic data for analysis (Renfro, 1980a).

AQ: The spelling “Mnemonic” has been changed to “mnemonic”. Please check.

The IBM Personal Computer at its introduction in 1981, with its original DOS (Disk Operating System) command line user interface, can be seen to be immediately compatible with the type of command line operation associated with the econometric modeling languages developed for use with time sharing mainframes in the 1970s. Furthermore the interactive operation of time sharing operating systems, which normally provided the context of the early development of such modeling languages, was functionally (if only very locally) mirrored by the single user operating systems of the microcomputer. Therefore, from the first, the microcomputer provided a new, yet also quite familiar environment. What this machine in addition soon made available to each user, beginning in 1982, was a pixel-based screen display that permitted graphical displays of a superior type that involved a matrix of points, in the form of pixels, that were individually addressable. Such a screen can be described as being “all points addressable,” rather than only line by line. Only rarely available previously to users of mainframe computers, this type of screen provided the environment for the development of the modern Graphical User Interface (GUI). Incidentally, the particular circumstance that caused the IBM Personal Computer and compatibles to be selected by almost all econometric software developers in the early 1980s, rather than the Apple, Tandy, or other microcomputers, may reflect the early availability for this machine of Fortran and other algebraically oriented compilers, in addition to the inclusion in its technical specifications of a numeric coprocessor chip, the 8087, which permitted faster floating point numeric calculations. For many years, the Apple machines, in particular, provided attractive frosting but almost no cake; with the exception of its stunning display, only in the present century has the Apple finally become hardware competitive with the PC.

Of course, the Personal Computer and modeling languages were independent developments, even if the microcomputer environment, taken together with the subsequent widespread use of this computer, caused a fundamental change in the degree of computer use worldwide. Considered alone, these modeling languages represent a logical extension of the development of high level programming languages that began in the middle 1950s. Both the parsed evaluation of alphanumeric commands and the translation of arithmetic/algebraic expressions, usually involving the conversion of infix notation (for example,  $a + b$ ) into reverse polish (for example,  $ab+$ ) or some other operative syntax that permits stack-based processing, constitute operations that are – or can be seen to be – common to both compiler design and econometric modeling languages. In turn, linker operation and the functional integration of a sequence of operations so as to marry the output of an earlier one to the input requirements of a later one are logically generally analogous in their essential characteristics.

During the 1970s, there was almost always a noticeable difference between the human interface of the econometric software packages typically used by academic economists and that experienced mainly by business and other nonacademic economists, who used the econometric modeling language type of interface just described. This difference in part reflects that, during the 1970s, batch processing mainframes and minicomputers were much more commonly available in academic environments than were computers with time sharing operating systems. The typical

self-programming academic econometrician in the 1970s might, in any case, have had little incentive to develop a sophisticated language interface for a program, compared to the incentive to focus upon econometrically interesting algorithms, but in a card (or paper tape) oriented batch environment there was even less reason. AUTOREG, B34S, LIMDEP, and most other such programs were originally developed with an algorithmic focus, rather than on the interface. Programs such as DAMSEL, EPS, MODLER, and XSIM were more human interface-biased in their development. The combination of differences in developer incentives and their environments explain the particular diverse characteristics and almost bipolar orientation of econometric software development during the 1970s.

However, it is also pertinent that, until about 1978, much of the design and development of econometric software occurred under relatively isolated conditions. There was a time in the early 1970s that journals, in particular *Econometrica*, appeared ready to publish articles and notes about software, but for whatever reason this was a short-lived, Prague Spring. With certain exceptions (Belsley, 1974; Eisner, 1972; Eisner & Pindyck, 1973), it was only at the end of this decade that program descriptions and algorithmic details noticeably began to appear in the disciplinary literature (Dent, 1980; Hendry & Srba, 1980; Kendrick & Meeraus, 1983; Kirsch, 1979; Lane, 1979; Pesaran & Slater, 1980; Society for Economic Dynamics and Control, 1981). Otherwise, econometric software and its documentation ordinarily passed from hand to hand, even if user guides to statistical programs had begun to appear in university bookstores. Of course, in the days before microcomputers, software purchases were commonly made organizationally, usually by people who worked in computer centers and spoke of “statistical,” rather than “econometric” software; in addition, it was decidedly uncommon for software of any type to be prominently marketed at the annual economic association and society meetings. Furthermore, even as late as 1983, computational methods were ordinarily investigated separately from any explicit consideration of their algorithmic computer implementation, and the citations that appeared in the formal economics and econometrics literature were often not directly related to any such implementation (Quandt, 1983), a practice not unknown today.

These circumstances of econometric software development before 1985 are relevant to the consideration of particular developments since. Furthermore, at the risk of stereotyping, it is useful to consider certain of the resulting properties of econometric software as the microcomputer began to be used widely, starting in about 1985. In particular, whatever the specific differences between programs in the 1970s, at that time it was almost universally characteristic of econometric software packages that they each offered specific, program dependent user choices. In the case of the econometric modeling languages, the user might be able to choose to create a possible variety of models, but the parameter estimation facilities were for the most part given. Some degree of flexibility might exist that would allow distinguishable techniques to be combined, such as Two Stage Least Squares and autoregressive corrections. Such packages also might offer greater capabilities to the degree an economist was willing to program, but essentially the typical user made his or her choices as if from a menu.

However, in 1985, a new type of software began to become available, the earliest example familiar to economists being Gauss (Hill, 1989). It is possible to argue that too sharp a distinction has just been made, that the econometric modeling languages already offered capabilities similar to those of these new packages, albeit described in the depths of thick manuals, but it is useful to ignore this particular fine point in order to focus on the difference *in orientation* of these two types of econometric software package. Packages such as EPS, MODLER, and XSIM are examples of econometric *modeling* languages (EML) (Renfro, 2004a, b, c, d), as has been discussed, but Gauss, Ox, and possibly other similar packages are effectively econometric *programming* languages (EPL). The critical difference is the object the user works with: an econometric modeling language characteristically has as its objects specific, well-defined econometric techniques, to include estimators with explicit names. Other objects take the form of time series variables, model equations, and models, but also a range of variable transformations, defined in terms of algebraic and arithmetic operators, and, as well, also implicit functions.

In contrast, an econometric programming language is defined by its mathematical and, in some cases, statistical objects. These objects include matrices, vectors, operators, implicit functions, a looping syntax, and a particular grammar, among other characteristics. As its name implies, an econometric programming language is a *programming* language, and one that is specifically oriented to the use of econometricians and economists. Generally, it is also a higher-level language than Fortran, C++, and other commonly recognized *computer* programming languages. An aspect of its high level nature is that the user is ordinarily not expected to be familiar with computer operating systems and other aspects of the particular use of a computer programming language. However, it is difficult to make hard and fast distinctions. Clearly, there is a potential classification question that could be raised concerning exactly how to distinguish an econometric programming language from any other programming language of a sufficiently high level. Similarly, as indicated earlier, an econometric modeling language can contain an econometric programming language as a sub category.

Suffice it to say that these are fuzzy sets. However, ignoring such categorical complexities, econometric software can today be classified into standard estimation packages that provide an economist with the ability to perform a given set of econometrically defined operations, operations that are specifically determined by the software developer. Notice that the operational characteristic in this case consists of the user *selecting* from a set of options, possibly using a menu. There is next a mid-range, which most obviously includes the econometric modeling languages, with the characteristic that the economist is required not only to make certain selections but also to determine how particular operations are performed: he or she must form equations, combining variables and operators and possibly implicit functions, and thereby *build* a model. These models can be *solved* or *simulated*. The results can be *plotted* or produced as *tabular displays*. Advanced Estimation Packages (AEP) or Generalized Estimation Packages (GEP) that both offer a selection of choices and incorporate a macro language capability should also be included in this classification, as offering a subset of capabilities and features. Finally, the econometric

programming language in turn offers less in the way of prefabricated statistical and mathematical objects, but more scope to *create* new econometric, statistical, and mathematical forms. It also might be possible to infer that an econometric programming language is most suited to use by econometricians, as creators of emerging techniques, as opposed to applied economists, who are more likely to use established methodologies, hence another type of package. Obviously, these sharp distinctions are most meaningful when considering polar examples of these package types.

Considering the human interface aspects of the modern econometric software packages, the classifications just described can be considered to imply substantial progress, inasmuch as the ideal might be to present economists with the capability to perform their research in the most immediately intuitively obvious way. For certain analysts, interested only in the use of standard econometric techniques, it is clearly beneficial for econometric software packages to be available that are easy to learn to use and involve little effort to apply. For others, the capability to learn an econometric language that is language compatible with the material presented in textbooks and journal articles would appear to offer much, even if this capacity might also imply the need to specify explicitly the calculations made in each case.

More generally, it might seem possible to infer from this description that this apparent movement towards a complete econometric programming language represents for economists what the development of CAD/CAM (Computer Aided Design/Computer Aided Manufacturing) has meant for architects, designers, engineers, and others, namely the creation of a productive environment in which it is possible to both design a new entity and at the same time constructively establish its specific characteristics. In an engineering context, the CAD component can be interpreted to permit the production of a design for a particular object; the CAM component ideally then permits the design itself to control the machine, or machines, that then produce this object. Alternatively, it might be possible to see these econometric software developments as implying potentially much the same type of near term functional improvement in econometric practice as modern word processing software has brought to document production, namely, in this case, a screen representation that is effectively the same visually as the final printed document, a characteristic that usually goes by the name *what-you-see-is-what-you-get*, or *WYSIWYG*.

All this sounds good at the outset, but there are certain aspects of econometric software that make these concepts less than immediately applicable. In the first place, in the case of econometric software, there is no necessity for there to be a direct correspondence between what appears on the screen and the computations that are made. Users of this software generally do not and will not know the algorithmic details of the computations performed, for the simple reason that individual developers do not ordinarily publish these details. Furthermore, whatever the user specifies in the form of a command, there is no necessary relationship between this command and the specific calculations algorithmically performed by the software. At issue here is not only the user's ability to specify the characteristics of a particular arithmetic or algebraic operation, but also the specific way in which various conditions are evaluated, such as, for instance, the manner of convergence in the

context of an iterative nonlinear process, or the user's freedom to set initial values and other control parameters (McCullough & Renfro, 1998, 2000).

Fundamentally, whatever set of commands the user provides will be *interpreted* by the software package, acting as an intermediating agent. The actual calculations then performed are determined and controlled in advance by the software developer. Obviously a choice that the developer can make is to permit the user's commands – whenever appropriate – to be implemented exactly as stated, but even this possibility is the developer's choice and therefore constitutes intermediation. The choice to allow the user algorithmic control is by no means necessarily the best. In at least certain cases, perhaps even most cases, it can be argued that it is desirable that the user of the package *not* be allowed to control precisely how the program does what it does inasmuch as that user cannot be presumed to be a knowledgeable numerical analyst, nor necessarily an experienced programmer who will also take the time to evaluate qualitatively the results obtained.

## Directives Versus Constructive Commands

In certain respects, the discussion has come full circle since the introductory section of Chap. 1. Recall the argument made there that, over the past 30–40 years, specialization has occurred, with the majority of economists effectively ceding responsibility for the design and development of econometric software to a minority of econometricians. In contrast, one of the implications of an aspect of the modern development of this software, namely the creation of econometric programming languages, would appear on the face of it to provide any enterprising economist with the effective capability (once again?) to design and develop his or her own software, but now in a way that avoids many complexities, yet achieves the goal of allowing that person to determine the constructive characteristics of whatever applied econometric research project he or she might wish to undertake. An objection that has been made to this idea is the argument just posed that, in any case, the designer and developer of any econometric programming language actually remains in control as an intermediating agent, whether this control is exercised or not. A normative question that naturally arises is, to what degree and how should this designer/developer actually exercise this control given the inevitable complexities of the computational process?

In order to address this question properly, a certain amount of background information is necessary. It is useful to begin by considering exactly what distinguishes a directive from a constructive command. A *directive command*, or simply a *directive*, as this term will be used here, can take any of a number of forms. For example, in order to direct a program to perform an Ordinary Least Squares regression of a named dependent variable, such as CE, on one or more other named regressors, the user might in one case issue the commands:

Dependent: CE

Regressors: YPD, CELAG1



in another:

$$CE = F(YPD, CE(-1))$$

or, in a third:

$$\text{Reg Command: } CE = c1*YPD + c2*CE(-1) + c3$$

Each of these types of directives are found in the command languages of one or more econometric software packages. It is also true that in some cases, pull down or drop down menus will instead be used in order to identify progressively the dependent and regressor variables.

All these directives are constructively equivalent, inasmuch as none do more than direct that a certain type of operation be performed. In all cases, the command's meaning and the particular corresponding default operations will have been established by the program's designer. That is, the meaning of the directive is completely established by the syntax and vocabulary of the program used. However, as illustrated, a directive can in some cases seemingly or even actually have constructive features; for example, notice that in the second command above, the term  $CE(-1)$  itself constitutes the directive that the variable named  $CE$  is to be retrieved from the program's data storage component and then lagged by one period before the observations on this variable are used as one of the regressors in the implied regression. In the third case, a linear-in-parameters regression specification also appears to be explicitly indicated. Nevertheless, notice also that none of the directives considered are, except by default, linked to a particular regression method.

In contrast, a textbook consideration of the general linear model and Ordinary Least Squares regression will commonly begin with a statement like:

Consider the linear regression equation:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{u}$$

where:

$\mathbf{y}$  – a vector of  $T$  observations on a variable  $Y$

$\mathbf{X}$  – a matrix of  $T$  observations on  $k$  regressor variables

$\boldsymbol{\beta}$  – a vector of  $k$  unobserved constant parameters

$\mathbf{u}$  – a vector of  $T$  unobserved disturbances

This opening set of definitions will be followed, by and by, with the statement that the ordinary least squares estimator is defined as:

$$\mathbf{b} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$$

If this operation were actually to be carried out constructively using pencil and paper, given an understanding of linear algebra and the availability of a particular data set, the most direct way to proceed is to compute first the sums of squares and cross products of all the relevant variables and then load these into a matrix. As is shown in almost any modern econometrics textbook (Davidson,

2000; Greene, 2003a; Johnston & DiNardo, 1997), and even in many older ones (Goldberger, 1964; Johnston, 1963; Theil, 1971), if this matrix is formed so that the cross-products of the dependent variable with the regressor variables border those of the regressor variables alone, a result is obtained that can be characterized as:

$$\begin{array}{cc} \mathbf{X}'\mathbf{X} & \mathbf{X}'\mathbf{y} \\ \mathbf{y}'\mathbf{X} & \mathbf{y}'\mathbf{y} \end{array}$$

Constructively, the next step is simply to invert the interior matrix,  $\mathbf{X}'\mathbf{X}$ . Following this inversion, the estimated value  $\mathbf{b}$  can then be computed by carrying out the matrix multiplications indicated by its above apparently constructive definition. Somewhere, in individual textbooks, at least historically, Cramer's Rule may be provided as a constructive definition of matrix inversion.

In contrast, if this estimation process were to be considered as a computer programming task, using some combination of a programming language such as Fortran or C++ and possibly Assembly language, the process of computing the estimates can instead be programmed so that, *as the computation occurs*, the right-most column of the original bordered matrix *simultaneously* becomes the location of the estimated values of the parameters (Goodnight, 1979), denoted by  $\mathbf{b}$ :

$$\begin{array}{cc} (\mathbf{X}'\mathbf{X})^{-1} & \mathbf{b} \\ \mathbf{y}'\mathbf{X} & \mathbf{y}'\mathbf{y} \end{array}$$

where  $\mathbf{b}$  is the set of estimated parameter values. The reason to make the calculations in this way, rather than to compute:

$$\mathbf{b} = (\mathbf{X}'\mathbf{X})^{-1} \mathbf{X}'\mathbf{y}$$

by making the implied matrix multiplications, once given  $(\mathbf{X}'\mathbf{X})^{-1}$ , is that such matrix operations, if carried out explicitly are in fact not efficient and may in addition result in greater rounding error, compared to the simultaneous generation of the inverse and the parameter estimates. However, as a matter of interface design, the program's developer is in no way constrained not to allow the program user to specify  $(\mathbf{X}'\mathbf{X})^{-1} \mathbf{X}'\mathbf{y}$  as a *directive*. Neither is the developer constrained to compute  $\mathbf{b}$  in any particular way, whatever the user's directive. But does it therefore follow that the developer should always act as a "correcting" intermediary?

In the beginning, specifically in the mid 1960s, it was common to find regression programs that replicated textbook calculations, as indicated in chap. 1. In those days, existing programs were commonly shared in the form of source code and an economist might therefore begin a research project by obtaining a deck of Hollerith cards onto which had been punched this source code. At first, because of time and effort constraints, it was natural in such cases to make only changes that had to be made and otherwise to leave well enough alone. However, in 1967, James Longley (Longley, 1967) evaluated a number of the existing statistical regression programs and discovered that, for at least certain data sets, they could be disastrously

numerically inaccurate, essentially because of the problem of ill conditioning and the use of single precision values during calculations. The possibility of this type of computational problem occurring had, in fact, been known to human computers at least as early as 1943 (Hotelling, 1943; Rushton, 1951), if not well before, but it had been forgotten. It is not actually particular to electronic computers, but there is no point now in considering it any more generally.

When the matter considered is the calculation of linear Ordinary Least Squares parameter estimates, or linear-in-parameter estimates more generally, it is possible to regard the fundamental threat to be the degree to which the data used are collinear. As discussed towards the end of Chap. 1, the problem in this case is essentially due to the ease with which the calculations can result in intermediate values that have no precision whatsoever, possibly implying the need at the very least to compute and provide the  $\mathbf{X}'\mathbf{X}$  matrix condition number as a potential warning (Belsley, 1991; Belsley et al., 1980). More generally, an aspect of the use of finitely precise numbers is that number comparisons can only discriminate between those that differ by a certain minimum amount. It is not meaningful to ask if  $x = y$ , but rather only if  $|x - y| \leq \varepsilon$ , where  $\varepsilon$  is some suitably chosen small number and  $x$  and  $y$  are floating point real values. One of the implications is that, even in the linear case, computations such as matrix inversion must be carried out with due regard for the effect of the data used, as a matter of conditioning, as well as the fact that in the end the solution is always *approximate* rather than exact (Higham, 2002; Stoer & Bulirsch, 1993).

This inexactness has a number of practical consequences. For example, to the mathematical economist, there is a sharp conceptual difference between a linear and a nonlinear problem. In contrast, to the econometrician in the guise of a numerical analyst, the environmental computational difference between a linear and a nonlinear problem can be fuzzy. It can be that the latter involves the need to make additional, more open-ended calculations in a context in which each successive calculation could progressively involve additional rounding and approximation error, although it is also true that nonlinear problems can involve specific computational issues, some of which arise from such things as the need to compute derivatives as finite approximations, local versus global maxima, initial conditions, and stopping rules (McCullough & Renfro, 2000). Recall that error propagation is not necessarily particularly serious in the case of multiplication, division, or taking square roots, but simply adding operands of different sign can, in extreme cases, lead to catastrophic cancellation (Stoer & Bulirsch, 1993, p. 11–12). In addition, rounding error can be local to a given iteration sequence in the case of convergent iterative calculations (Ralston & Rabinowitz, 2001, p. 334). The relevant issues and aspects are varied, but in the end what fundamentally needs to be understood is that infinitely precise calculations do not occur within an electronic computer, and that the name of the game is the minimization of calculation error, not its absolute elimination.

When considering these matters, notice also that the devil is in the details. For instance, when considering the textbook expression

$$\mathbf{b} = (\mathbf{X}'\mathbf{X})^{-1} \mathbf{X}'\mathbf{y}$$

it is tempting to interpret it initially as being essentially constructive in nature – in part because of its familiarity to econometricians – but if the sequence of elementary calculation steps are set out carefully certain inherent ambiguities can become evident along the way, especially to the degree that  $k$  regressor variables are considered, rather than 1, 2, or even 3. For instance, there is the issue of the best way to invert  $\mathbf{X}'\mathbf{X}$ , as well as the precise way in which other calculations are made, not excluding the construction of the matrix  $\mathbf{X}'\mathbf{X}$  itself. The precision with which each of the numbers are calculated needs to be considered – and also the precision with which they are stored at each stage of the computational process. There are numerous opportunities to make serious errors in the calculations that might not be noticed at first, some of which can be due to the order in which the individual calculations are made. And if the final results are presented without the clear identification of the specific numbers used as original inputs to this computational process, it may be difficult for someone else to validate the results later, even given knowledge of each computational step. This example of course represents a relatively simple case in this day and age.

The original consideration of the numerical accuracy of regression programs by Longley (1967) brought to the attention of econometric software developers, among others, the problem of rounding error in the context of single precision floating point numbers. During the intervening years, there has been a significant amount of work done concerning the numerical methods that should be adopted, most recently considered by Stokes (2005), who also addresses data storage precision, as well as alternative matrix inversion techniques. The particular methods of matrix inversion that should be employed, generally speaking, are determined by the characteristics of the data: in particular, near singular matrices imply the need to dispense with the usual Cholesky factorization of  $\mathbf{X}'\mathbf{X}$  and to use QR decomposition applied directly to the data matrix. However, before performing a regression, it is commonly not evident just how collinear the data are, which once upon a time created a conundrum: in earlier years, in the 1960s – in the case of mainframes – and the 1980s – in the case of the microcomputer – there was an issue concerning the demands accurate matrix inversion techniques placed upon the capabilities of existing CPUs as well as computer memory. Today, it is generally no longer necessary to worry about this as a matter of computer resource cost: few techniques likely to be employed will today require more than a literal fraction of a second, given the use of a modern computer, and high speed memory has become comparatively abundant. But even if the stakes associated with “capital intensive” methods of calculation are no longer what they once were, it is still true both that a design decision must be made and, if the wrong decision is made, that it may not be obvious to the computer user whenever the calculations involve excessive errors. A classic consideration of the computational problem of error accumulation is that by Harold Hotelling (Hotelling, 1943), but see also Wilkinson (1961), Belsley et al. (1980), and most recently McCullough (2004) and Stokes (2005). For a consideration of the algorithmic properties of certain simultaneous equation estimators, see for example Kontoghiorghes et al. (Foschi, Belsley, & Kontoghiorghes, 2003; Kontoghiorghes, 2000; Kontoghiorghes & Dinienis, 1997).

At present, it is not necessary to consider in further detail the particular numerical analysis issues associated with econometric computation, for the purpose of this exposition is not to identify and catalogue the specific provisions that need to be made for numerical accuracy in each case. The aim is rather to make clear that how calculations are performed does matter and that the provisions made are relevant to the way econometric theory is practiced – or should be. Numerical analysis issues need to be addressed not only behind the scenes, when the design and development of econometric software is specifically considered, but also in the mainstream literature when the putative properties of estimators, diagnostic tests, and other econometric topics are discussed. The reason is simply that the characteristics of the computations made can affect the validity of the inferences that are drawn by those who apply theory to the real world.

However, in addition to the operational need to produce sufficiently accurate results, there is also the matter of computational completeness, briefly discussed earlier. Textbook and journal presentations, jointly and severally, inevitably provide only partial coverage of the range of formulae and calculations that are pertinent. Implementing Ordinary Least Squares, or any other parameter estimation method, so as to produce a program that is generally reliable in its operation requires information that is sometimes difficult to discover: for example, what calculations to permit in a variety of special cases, such as when the program user chooses to suppress the constant term, which among other things affects the supplementary statistics that are ordinarily displayed next to, above, or below the parameter estimates? Alternatively, how should the program react if the user chooses to regress a variable on itself, or omits entirely all regressor variables? Or, what if the data set used exhibits missing values, either at the extremes or for observations in the interior of its range? The relevant consideration here is not that these events are each necessarily the result of sensible actions on the part of the program user, but rather that particular actions, if not dealt with properly, can cause the program to crash, or perhaps to produce (possibly without warning) results that are neither correct nor appropriate under the circumstances. Ideally, *any* action that the user is both permitted and chooses to make should result in a meaningful and graceful response on the part of the program, taking the form either of an intelligible error message or a sensibly determined result.

Still other aspects of the econometric computational problem could be considered. However, it is already self-evident, from the perspective of software design, that many (most?) of the seemingly constructive formulae that populate the econometrics textbooks and the general econometrics literature should in fact be considered to be simply directives. Furthermore, it is clearly quite justifiable to argue that, ideally, those econometricians who create the software used by economists generally – both in the case of standard packages, that involve user selection of options, *and* in the case of econometric programming languages and other contexts in which seemingly constructive commands might appear – should carefully take into account what the translation from the mainstream econometrics literature to software embodiment implies. However, it is not always immediately evident how this translation should be effected. Giving the typical economist what might be appear to

be full control of the operations performed, in the case of an easy-to-use econometric programming language, yet monitoring carefully every command – so as to insure numerical accuracy, among other considerations, as well as to permit program control errors to be trapped and then dealt with gracefully – would seem to place the econometric software developer in a courtesan's role, not only assuming, in the famous phrase, “power without responsibility,” but also flattering the user into thinking of him or herself as having accomplished something dangerously difficult or at least deeply satisfying, yet all the while with the training wheels still firmly attached.

But is it reasonable to suppose that econometric software of any type, including econometric programming languages, can be created so as to be quite so foolproof to use? An obvious aspect of the matter being considered here is the question whether the programming of econometric computations can ever be simplified in a way that easily permits any type of relevant calculation to be performed and insures the accuracy and completeness of the results? Very early in the historical development of the electronic computer it became apparent that the availability of labor saving “building blocks” would be helpful as programming aides, which lead to the development of subroutine and function libraries (Wilkes et al., 1951). It was evident that, ideally, the creation of such libraries as collections of elemental computational “tools” might both simplify the programmer's task and simultaneously provide algorithmic components that would presumably meet appropriate qualitative standards. One of the beneficial effects might be to permit programs to be created using prefabricated “parts,” more or less in assembly fashion. The development of econometric programming languages can be viewed as simply a further evolution of the idea of providing any person wishing to perform calculations with a reliable set of tools from which a selection can easily be made. However, the collective programming experience of now nearly 60 years has shown that the assembly of computer programs can only be simplified to a certain degree, still requires the assembler to acquire skills, and implies the need for an information transfer to occur – in both directions – between the tool builders and the tool users. It also true that even if the programming process itself can be simplified to the point that each individual “tool” is separately easy to learn to use, the inevitable multiplicity of them is likely to make the learning process time consuming. Furthermore, the subsequent application of these “tools” to the applied economic research process involves complexity on this score as well.

## **Developers, Users, and Use**

Certain implications are obvious and particular questions have been posed. It is now time to consider directly the \$64,000 question: what then is there about economic or econometric computation that permits the argument to be entertained that, after many years of neglect, the production and use of econometric software are activities that should be consciously regarded as being fundamentally within-discipline,



not only that, but as calling for open discussion in a disciplinarily central place? Is it not enough that economists should be content to use this software? Have its properties not already been evaluated enough? It is, of course, true that software reviews have appeared in the econometrics journals, actually more than 120, based upon the count by McCullough and Vinod, which they report in a recent issue of the *Journal of Economic Literature* (McCullough & Vinod, 1999). Unfortunately, almost all these have tended to superficiality in their treatment (McCullough & Vinod, 1999). In particular, among the defects of these reviews, only four considered the numerical accuracy of the software (Lovell & Selover, 1994; McCullough, 1997; McKenzie, 1998; Veall, 1991), just a few attempted to address the comparative suitability of different programs for particular applications (Mackie-Mason, 1992), and some fell short of being independent evaluations. Otherwise, as intimated earlier, to the degree that economists or econometricians have overtly considered the subject of econometric software, there has been a pronounced tendency to treat its characteristics as being at best of very limited disciplinary relevance. Except in exceedingly rare cases, econometric journal articles and textbooks noticeably have not discussed, indeed have seldom ever referred to, any substantive software design or algorithmic issues. Furthermore, as will be demonstrated, when applied research results have been presented, involving computational aspects, the particular software that may have been used to generate those results has seldom even been mentioned.

### *Use and Users*

What can in fact be discovered fairly quickly about the revealed software preferences and use characteristics of the typical economist? The answer is, a few things, which derives from the existence of JSTOR, which has been developed as a full-text-searchable online archive of journals and is of course Internet-based. Among other things, JSTOR permits at least a portion of the economics literature to be keyword searched for such terms as “software package,” “computer program,” “econometric software,” or even for the names of particular software packages, in order to examine the degree to which economists describe their software use in the context of published articles, including for example possibly the extent to which evaluative tests are carefully and knowledgeably used. Welcome to Bibliometrics 101; however, notwithstanding any appearance to the contrary, it is here being taught by a novice.

To start, there are certain important caveats. The first relates to the fact that JSTOR is made available to research libraries not just as a complete archive, but also as separable groupings, including economics journals, business journals, statistics journals, biological sciences journals, and so on. Although a large library might subscribe to all the groups, smaller libraries or specialist libraries can instead purchase selective access. One of the JSTOR characteristics is that these groups are not mutually exclusive, but instead, in certain cases, intersecting sets – as is in fact the case for the ostensibly separate economics and business journals groups. Therefore,

it is not necessarily meaningful simply to contrast search results obtained from the business group with those from the economics. Another limitation is that the publishers of most journals impose a multi-year moratorium on the inclusion of their most recent volumes in this archive, thus establishing a “moving wall” that year-by-year inches forward and may vary from journal to journal and group to group; for a group, the results for the most recent years, going back as much as 5 years, may exclude volumes for particular journals. In addition, as an operational characteristic, as everyone knows from Internet search experience, keyword searches involve potential inferential pitfalls. For example, taken at face value, approximately 26,000 articles published during the mainframe period from 1960 to 1984 appear to include at least one mention of the program “GIVE” (or at least words consisting of various mixed cases of this specific sequence of letters, being the name of the predecessor of PcGive). This is no doubt a heart-warming thought for David Hendry, especially as, in contrast, other programs, such as EViews, MicroFit, MODLER, and TROLL, either individually or collectively, seem to be mentioned in only 330 articles during the entire period from 1960 through 2003, notwithstanding the possible appearance of rEViews. Finally, the particular results obtained in each case can also be affected by specific settings of the “Advanced Search” criteria, including “search full text content only” and limitations of the search to “articles” only, versus the alternatives. In addition, the number of journals included in JSTOR is increasing, so that the results reported here might not, even now, be able to be exactly duplicated. However, the good news is that anyone with access to JSTOR can play.

If due allowance is made for the effect of the use of common words, and for the fact that one person’s “package” is another’s “program,” what is easily discovered is that searches using the terms “software package,” “computer package,” “computer program” and “econometric software” together yield a total of less than 2,400 articles in which any of these terms appear during the 64 years from 1950 to 2003 – for this specific collection of terms, a total of 2,395 “hits” are obtained, the first in 1958. This number is reasonably large, but a subsequent article-by-article examination of a reasonable sample of the “hits” obtained reveals that this number represents a substantial overstatement of the degree to which economists have either identified any software used or revealingly discussed their use of the computer during the years since 1960 – especially if only marginally evaluative software reviews that appear as “articles” are excluded (McCullough & Vinod, 1999). Restricting the search to “Articles” alone reduces the total to 1641. Adding “Reviews” increases the total to 1851. The further addition of “Editorials” brings the total to 1853. The number of mentions in the “Other” category alone is 542. It is easily, if tediously, determined that the vast majority of the articles discovered by the search discuss economically relevant aspects of the use of the computer by economic agents or else aspects of the economic role of the computer during these 50 + years, but – for the most part – *not* how economists use the computer.

But if dreams have been shattered, there are still some interesting results. Considering the JSTOR findings in more detail, in the 52 journals identified there as “economic journals,” it appears that from 1960 to the end of 2003, a total of 2,176 articles were published in which the word “software” occurs at least once. The choice

of this particular ending date is made because of the “moving wall.” The first article in which the word “software” is used was published in 1962 and the second in 1965 (Diebold, 1962; Levy, 1965). The first 109 of these articles were published before 1979, with 122 of them published in the next 6 years; hence, all but 231 have been published since the beginning of 1985 – the first to be published in 1985 (January) happens to be a review entitled *Econometric Software for Microcomputers* that with the exception of MicroTSP (now EViews) and SORITEC focused entirely on *statistical* software packages. Looking further back, the first use of the term “computer program” occurred in 1958 (Moore, 1958), although the first related use of both “computer” and “program” in the context of an article occurred in 1955, appropriately enough one written by Herbert Simon (Simon, 1955). In the 1,098 times “computer program” appears at least once in an article before 1 January 2005, the first 446 occurred before 1981; the last time it appeared was in October 2003. The alternative term “computer package” scores 76 “hits”, the first in 1976 and the last in 2003. Only 182 articles contain the more specific phrase “econometric software,” arguably the most widely used category of “economic software,” although this seemingly more global term itself actually receives a total of only 1 hit. “Econometric software” is of relatively recent coinage: of the articles mentioning it only 14 were published prior to 1987, the first one by Robert Shiller in 1973 (Shiller, 1973). In contrast, 94 have been published since 1995. Perhaps surprisingly, inasmuch as it is commonly considered to be a broader category, “statistical software” receives fewer hits, only 100, with the first article containing this term appearing in 1982.

Anyone who wishes to luxuriate in the innate practicality of economists can take heart from the fact that the composite phrase “theoretical software” is absent from the JSTOR economics database – or does this instead indicate something else? From 1960 through 2003, a total of 60,202 articles are found in which the word “theoretical” or “theory” (or both) appear least once, 55,234 in which “theory” appears, 31,709 for “theoretical,” and 26,741 in which both these words appear. In contrast, “software” appears without “theoretical” or “theory” in only 1,136 articles. In order to determine the population of all English language articles published from 1960 through 2003 in economics JSTOR journals, a search for all the articles that contain the word “the” (which can be given the acronym ACTWT) results in 83,659 hits, which essentially provides a database population count, since surely no article can be written in English without using this word. This finding suggests that essentially 72% of all articles may have some theoretical content. On the other hand, the word “data” appears in 52,085, which is 62% of all articles. Unless this finding implies only that data is something economists like to theorize about – from 1960 through 2003, 39,136 articles were published in which “data” and either “theory” or “theoretical” appear – it is seemingly something of a mystery what exactly was being done with all that data.

The combination of these findings with other, independent use-evidence lends support to the idea that economists have often employed econometric software without much mention of that use. Given that people ordinarily use what they pay for, an indication of a significant degree of actual use is the reasonably substantial revenues collectively generated by econometric software programs such as EViews, MicroFit, MODLER, PcGive, and TROLL, in contrast to the approximately 190 combined

hits obtained searching these names. This hit count is at least a slight overestimate, inasmuch as it might include such obviously noisy results as would be implied by articles on such things as collective bargaining in Pacific coast fisheries, because of the verb “to troll;” furthermore, none of these programs existed prior to 1968 and the names of certain programs date only to the 1980s or later. Restricting the time period to 1986 through 2003 results in a hit count of 130. Even TSP, which is apparently the individually most mentioned econometric software program (with 397 hits for the period between 1965 and 2003) and one of the longest existing, is subject to false positives, despite being a made-up name, inasmuch as, for instance, a 1967 article on the Theory of the Second Best attracts a hit, possibly because of a misprinted TSP for TSB.

However, economists do not live by econometric software alone. Examining the hit rate for the statistical software packages SAS and SPSS yields 632 hits for SAS and 132 for SPSS for the period from 1975 through 2003. The group that includes EViews, MicroFit, MODLER, PcGive, SAS, SPSS, TROLL, and TSP yields a total of 1,273 hits for the same period, which may imply a tendency for economists to use SAS or SPSS rather than any of the econometric software programs. It has previously been suggested (Renfro, 2004a, b, c, d) that SAS, SPSS and other statistical software packages were used disproportionately by economists during the mainframe era, in part because computer centers often leased software likely to be used by a broad range of disciplines. The fact that beginning in or about 1985, economists for the first time potentially could choose their own software, makes it interesting to consider whether any differences can be discovered between the time before 1985 and since. As it happens, if the search time period is limited to the period from 1985 through 2003, the econometric software packages on their own achieve 430 hits. Adding SAS and SPSS to the collection raises the number of hits to 1,034. Either SAS or SPSS (or both) are mentioned in 630 articles published from 1985 through 2003. Without trying to be too precise, it appears that economists were as likely to mention SAS or SPSS in their articles in the post 1985 period as before that time. From 1975 through 1985, SAS or SPSS achieved 148 hits; from 1986 through 2003 a total of 601 hits. What this finding implies about relative usage, and why it is that economists may have continued to use SAS and SPSS (dis)proportionately just as often since 1985 as before, are potentially interesting questions.

It is additionally potentially informative to combine in the searched population for the period from 1960 through 2003 the articles in the 23 journals currently identified by JSTOR as “statistics” journals. In this case, the number of articles found containing the word “software” increases to 5,619, and those containing “computer program” to 3,023. The term “econometric software,” in contrast appears in only 196 articles in the combined population, obviously implying that only 14 articles in this particular “statistics” literature sample contain a mention of this term. On the other hand, “statistical software” achieved 748 hits in the combined sample, nearly eight times the number in the economics journals alone; the first affected article in the statistics literature was published in 1970, in the form of an article on statistical training and research (Patil, Box, & Widen, 1970). In the combined sample, 12 articles mentioned “useful software,” but these were mainly in statistical journals; in economics journals articles, only Zvi Griliches and Paul Romer felt

this usefulness strongly enough to mention it, other than those economists who apparently used this phrase in connection with the use of computers by Ohio commercial farmers or when dealing with the use of simulation models in the classroom. Anyone with access to JSTOR can easily generate additional results in order to examine the implications of other word combinations and to probe other sample configurations.

Of course, it is also true that even if economists had always been particularly careful to document their computer use in their published articles, the JSTOR results would still only provide a partial result, for the simple reason that the journals included are those that ordinarily publish the work of academic and academically oriented economists, who may also work in government and other research-focused contexts, but generally not that of business economists and others less likely to publish in these journals. Those less likely to publish in such journals nevertheless constitute a significant part of the market for econometric software. Packages like LIMDEP, MicroFit, PcGive, and SHAZAM, on the one side, and AREMOS, AutoBox, FP, MODLER, Stata, and TROLL, on the other, both categorically and individually appeal to distinct user groups and types among economists. For instance, academic economists probably use programs in the first group disproportionately. Some of these users may also have particular geographic characteristics: MicroFit and PcGive, for example, may be more likely to be used in the UK and Europe than in North America. In contrast, AutoBox, with ARIMA as a specialty focus, tends to be most used by business economists and others who would describe themselves as forecasters. AREMOS, FP, MODLER, and TROLL are most likely to be used by those interested in working with macroeconometric models, with FP and TROLL of perhaps greatest interest to those interested in simulations that incorporate the phenomenon of model-consistent expectations. Stata appeals to both academic and business economists, but within the latter grouping not those who primarily consider themselves forecasters.

Actually, remember that JSTOR has a category of business journals, in addition to economics journals, but as discussed earlier these are not mutually exclusive categories, nor is the journal selection in each category such that it would be possible to separate the academic from business economists by journal without making individual selections. For example, taking a simple minded approach, adding the business journals to the economics obtains the result that “econometric software” receives a total of 222 hits. Among the business journals alone, this phrase receives 217. Searching for a mention of any of the terms “EViews, MicroFit, MODLER, PcGive, TROLL” receives 221 hits among the business journals, and 226 when the business and economics journals are combined.

It is also possible to consider the research practices of others. What provides both an interesting and perhaps telling contrast to all the results just described is that if the 83 biological sciences journals in JSTOR are searched just for “SAS” for all its history, 1975 through 2003, there are 11,519 hits. If “SAS” and “SPSS” are jointly searched, for the period from 1975, the number of hits is 14,626. The ACTWT population count is 326,747. Most importantly, once these searches are made, although an article-by-article inspection reveals some false positives, it also reveals

that a vastly higher percentage of the time an empirical research article is “hit,” the software package being used in the reported research is explicitly identified within it, as compared to the case of econometric software packages and the economics literature. What an article-by-article inspection also reveals is that authors often report not only the software package used but also the specific procedures employed, the software version, and other significant details. Obviously, this quick comparison “test” has its flaws. For instance, it does not reveal how often those who contribute articles to these biological sciences journals omit any identification of the software used in their empirical research, particularly once adjustment is made for this type of article versus any other, or the effect if the full range of packages that potentially could have been used were to be included. Furthermore, no comparison has been made of the number of empirical research articles published in each case. Nevertheless, there is a possible object lesson here for economists.

The implication would seem to be that, among economists, econometric software design is commonly held to lack interest or relevance as a disciplinary topic and, furthermore, that it is generally thought that the use of a particular econometric software package or a particular version, rather than another, is of no material consequence. The conventionally accepted role of econometric software appears to be, as asserted in the introduction to Chap. 1, simply that this software makes operational, in an essentially slavish fashion, the formulae and related results that are to be found in the existing printed literature. As a consequence, certainly throughout the twentieth century, a reader of the economics and econometrics literature might well conclude that the development of this software has had no perceptible effect on the development of economics or econometrics other than to make faster calculation possible, essentially as a result of the speed of the computer as compared to alternative methods.

A possibly revealing sidelight on this particular conception of this software is provided by a short comparative description of a recent version of an econometric software package written by an economics graduate student and teaching assistant. The description was intended to be an evaluation of the salient econometric features of two quite different programs and in its entirety took the form:

To be honest, I found Y’s interface to be very 1990s (and early 1990s at that) and 16-bit retro. The browse features with files in 8.3 with tildes in a land of long file names is pretty old fashioned. Compared to X, the product is very clunky. No right click context menus and a very non-intuitive interface. I really only spent a few hours playing with the software before I went back to X. So I really did not explore very deeply as I found the browsing and non-drag and drop environment a real handicap compared to other products. In my opinion, the product really needs a complete overhaul to make it competitive in today’s environment.

The writer exhibits no awareness that it was only the mid 1990s before Windows even began to be the common interface for econometric programs, and that it was only with the introduction of Windows 98 that right-click context menus first arrived as a standard feature – and that, because of its operating system origin, context menus are actually a feature of both programs X and Y, but perhaps not for all program features. This review makes no reference to numerical accuracy or other substantive issues. It is obviously concerned entirely with only certain aspects of



the human interface, including an apparently requisite drag and drop environment, in a way that suggests that the earlier discussion in this chapter of macros and other algebraic language matters would be quite a foreign idea to this resident of the “land of long filenames.” However, fundamentally, the question that this review raises is why the writer omitted altogether to analyze what each of the programs do, in comparison with each other. Perhaps the assumption the writer made is that below the surface they would be the same functionally, implying a consequent belief that the focus of any econometric software review should simply be the immediate intuitiveness of the interface.

The formation of this type of mindset during the late second half of the twentieth century and the first few years of the next appears to be a consequence of the circumstance that during the earlier years of this 50 + period those econometricians who developed econometric software did so in the context of their own applied research, or perhaps when acting in a research assistant role, using the electronic computer simply as a replacement for earlier computational technologies, hand or electromechanical. At that time, as discussed earlier, what first tended to occur was simply the application of well-known formulae in an essentially straightforward fashion, at least until it became known, essentially beginning in the later 1960s (Longley, 1967), that such an approach can result in highly inaccurate results. Of course, inasmuch as at this point in time the applied economist or econometrician characteristically self-programmed, the modifications required to ensure numerical accuracy could be made silently, with little need to communicate to others either how the particular algorithms used differed in their numeric analytic characteristics from econometric textbook formulae or the nature of any particular modifications. At that stage economists also hardly needed to tell themselves whenever they discovered that previous versions of their software exhibited specific computational faults.

However, by the middle 1980s, because of the widespread adoption of the microcomputer by people who had never programmed, most computer users became instead computationally vicarious, therefore ordinarily not particularly conscious of the specific computational techniques employed and seemingly all too ready to accept at face value whatever their chosen software provided. To be sure, there has always been some level of background recognition that it was possible for programming mistakes to be made, and economists have certainly been aware that such things could happen, but, except in the context of specialist works read almost exclusively by the computationally committed (Belsley, 1991; Belsley, & Kuh, 1986; Belsley et al., 1980), to date there has been a distinct reticence to consider openly this possibility and its implications (McCullough, 1997; McCullough & Vinod, 1999; Renfro, 1997a, b). A willingness to tolerate operational secrecy has long been characteristic of applied economic practice and, notwithstanding certain disquieting revelations from time to time (Dewald, Thursby, & Anderson, 1986; McCullough, Renfro, & Stokes, 2006; McCullough & Vinod, 1999), or occasional avuncular public scoldings (Griliches, 1985, 1994; Leontief, 1971), only recently have the most prestigious economics and econometrics journals, in their article submission requirements, noticeably begun to mandate the more careful

reporting of the data used and of relevant computational details (Anderson et al., 2007; Bernanke, 2004; Renfro, 2006). But only a few have yet implemented such policies and even fewer have taken the necessary steps to insure their effectiveness (Anderson, 2006).

Each of the circumstances just mentioned – the possibility of software “bugs,” that the formulae that appear in the econometrics literature may not exactly correspond to the specific computations performed by econometric software packages, and that software users and developers are normally different people – are each individually sufficient to establish the need for more public discussion of the characteristics of this software. In combination, they overwhelmingly establish this need. However, it is also true that various, somewhat more general, software design characteristics can also be shown to affect the success with which economists conduct their applied research, as well as the specific numerical results obtained (Renfro, 2004a, b, c, d). But what may have curtailed discussion among economists of the design implications is, as much as anything else, likely to be a general perception of the lack of any feedback effect from the process of software development to the development of either economic or econometric theory. Notice the specific wording here. It is not suggested that there has been no feedback, nor that this has been minimal, but rather that it has not generally been perceived, which may well be the result of the necessity to comprehend first how the design and development of econometric software can affect both applied economic research and the development of economic and econometric theory, before it is possible to perceive either the existence of that effect or its particular magnitude. Omitting to look can itself diminish perception. The Pollyanna problem that such indifference poses is that it is usually folly both to fail to encourage the development of those things that affect well being, yet still to depend upon and expect a successful outcome.

However, the winds of change are picking up. For example, quite recently, an article has been published in *Econometric Theory* (Kristensen & Linton, 2006) that proposes a particular closed form estimator, the need for which is there declared to be a direct consequence of reported, possibly inherent, computational problems encountered by econometric software developers when using standard numerical optimization procedures. This is a potentially interesting example of the type of coordinated investigation that could and should occur as a matter of course. However this degree of recognized symbiosis between the development of econometric theory and econometric software is nonetheless still exceedingly rare, with the development of theoretical econometrics so far seemingly only minimally inspired by workaday computational experience and, in addition, with little attention being paid by theorists to how *best* to implement their work computationally.

### ***Econometric Software Developers***

In all the preceeding discussion one actor has so far appeared in what might be perceived to be a shadowey, if not furtive role, namely the econometric software

developer. Actually, this is not a dishonorable calling, nor does it require anonymity. There may be little need at this stage to proclaim the names of the culprits, one by one, for these are available in the published compendium and in the manuals and references cited there (Renfro, 2004a, b, c, d), but something about them should be said. The activity is interesting for the types of econometricians it attracts and now includes. It includes economic and econometric journal editors and associate editors. It includes textbook writers, theoreticians, and applied econometricians. Many of its members are academic economists. Others are employed professionally to design and develop econometric software. By motivation, it may be an avocation, an addiction, or even a hobby. But there is also the question, should this group be regarded as including only those who happen to be active today, and not those who have played a part in the past? If its ranks are expanded retrospectively, to include those who during their careers have spent a reasonable amount of time programming, using dyed-in-the-wool computer programming languages, the category of econometric software developer is arguably even quite distinguished in its membership, a secret army consisting of hundreds, perhaps thousands of economists, notwithstanding that most may have given it up upon receipt of their last earned degree. However, as a active grouping, it may not be numerically self-sustaining. This is potentially a troubling idea. But hold this thought for a while, for there is yet more to consider.

The Practice of Econometric Theory  
An Examination of the Characteristics of Econometric  
Computation

Renfro, C.G.

2009, XVI, 311 p. 28 illus., Hardcover

ISBN: 978-3-540-75570-8