Econometric Software: The first Fifty Years in Perspective*

Charles G. Renfro
601 West 113th Street, #12G, New York, NY 10025
Email: cgrenfro@modler.com

This paper introduces a special issue of the *Journal of Economic and Social Measurement* on the development and evaluation of computer software for econometric applications. It describes at least certain aspects of the history of the development of this software during the past approximately 50 years, beginning with the first use of the programmable electronic computer by economists in the early 1950s. It considers the various types of software developed, ranging from packages that permit the user to select from a particular set of options, to those that form essentially an econometric modeling language, to those that offer econometric programming capabilities, potentially allowing the individual specification of the characteristics and properties of the operations performed. This paper provides a relatively extensive list of references and is supplemented by a separate compendium of existing econometric software packages.

1. Introduction

When we look at an acorn, it is difficult to imagine the tree that it might become. In 1933, in his introduction to the first issue of *Econometrica*, Ragnar Frisch famously characterized econometrics as

> by no means the same as economic statistics. Nor is it identical with what we call general economic theory, although a considerable portion of this theory has a definitely quantitative character. Nor should econometrics be taken as synonymous with the application of mathematics to economics. Experience has shown that each of these three view-points, that of statistics, economic theory, and mathematics, is a necessary, but not by itself sufficient, condition for a real understanding of the quantitative relations in modern economic life. It is the unification of all three that is powerful. And it is this unification that constitutes econometrics.

This depiction does not provide an accurate prediction of the subsequent development of this subject. In particular, whereas there has been intensive focus over the years on the properties of parameter estimators and much work done to frame parameter-related hypotheses tests, there has not been nearly the same degree of attention paid to "this unification." Quantitative economics, as the broader area is often termed today, includes a number of topics, such as index numbers, activity analysis, and economic measurement generally, that are seldom addressed in any modern econometrics textbook. The techniques of

* The historical development of econometric software occurred sufficiently recently in the past that most of the developers are still alive. This article benefits both from correspondence with a number of them and from having been circulated widely and from the comments received. I am particularly grateful to Gerry Adams, Terry Barker, David Belsley, Jon Breslaw, Allin Cottrell, Clint Cummins, Kenneth Berk, Tom Doan, Jurgen Doornik, James Davidson, Mark Eisner, Ray Fair, Bruno Giuseppe, Arthur Goldberger, Richard Goldstein, William Greene, Bronwyn Hall, Robert Hall, Stephen Hall, Tim Harrison, David Hendry, Peter Hollinger, Charles C. Holt, Lawrence Klein, Robert Lacey, Cynthia Latta, Edward Leamer, David Lilien, James MacKinnon, Keith May, the late Michael McCarthy, Michael McCracken, Bruce McCullough, Marc Nerlove, Ray O'Brien, William Peterson, Peter Phillips, Richard Pierse, Robert Pindyck, David Reilly, Colin Rose, George Schink, Ronald Schoenberg, Lucy Slater, Richard Startz, Houston Stokes, Daniel Suits, William Teeters, Kenneth White, Mike Wickens, Vince Wiggins, Mark Wiley, Clifford Wymer, Michael Young, and Arnold Zellner for their patient willingness to assist me. This account should nonetheless be regarded as a memoir, rather than history, both because it is written by an actor in the process, and because there are no doubt others who have developed such software and whose contributions are not recognized here. I am wholly responsible for all errors of fact and omissions, and for the opinions stated.

experimental economics are similarly ignored, and the orientation of econometrics implicitly remains the estimation of parameters using time series data, even if there is nowadays increasingly consideration of panel data and related statistical techniques. On a purely nominal level, it can of course be argued that to call the broader range of topics Quantitative Economics and the narrower issue of parameter estimation Econometrics is simply semantics, but it is hard to argue that during the next 70 years econometrics should not further evolve.

To a degree, the present state of econometrics reflects the research tools historically at the disposal of economists, including both the availability of data and the computer. However, the issue is not just when the computer began to be used in applied economic research, but rather when it became ubiquitous. Although "born" in the 1940s and used first by economists during the early 1950s [172], as a general phenomenon the programmable electronic computer became an economic research tool only during the 1960s.[1] A major part of this use was to support the creation of macroeconometric models, obviously reflecting the scope and funding of such projects. Computing required first the creation of software, and even more generally involved a steep learning curve, not to mention access to a computer. Robert Hall remembers that when he arrived at MIT in 1964, "there was an IBM 1620 in the basement of the Sloan Building, but no usable econometric software" [141]. To Hall, who by the spring of 1965 had developed his first program, this lack was a challenge. Similarly, in the early 1960s at the University of Wisconsin, Arnold Zellner and Arthur Stroud created programs to implement Seemingly Unrelated Regression Estimates and Two and Three Stage Least Squares as a by product of the development of these concepts [350-352].[2] At the London School of Economics, Mike Wickens submitted as his M.Sc. thesis in 1964 a Fortran program based on James Durbin's FIML algorithm [82] that, among other things, utilized Newton-Raphson convergence and demonstrated that the second iteration of the process generated Three Stage Least Squares estimates [342].[3] In April 1965, Michael McCracken's DATABANK system was in operation on a CDC 1604 at Southern Methodist University [213]. At about this time, in Auckland, New Zealand, a succession of Rex Bergstrom's graduate students began to create linear and nonlinear, single-equation and system estimator regression programs on another IBM 1620 and later an IBM 1130 [262, 263]. But these are exceptional cases. Large research projects were best able to incorporate and support such activity, inasmuch as even until the mid to late 1980s any project with computational complexity involved considerable time and effort, apart from conceptual design issues. Contemporaneous descriptions of model building projects undertaken during the 1960s and well into the 1970s commonly presented this process as necessarily involving 10 to 15 economists and research assistants for time periods of as much as a year or more, in order to construct a model of 100 to 200 or more equations [186, 216]. In the 1960s and 1970s, the reasons for such resource requirements included the lack of easily accessible data as well as

[1] There are a variety of reasons that 1960 is more or less the date at which the mainframe computer began to be used widely, but not the least of them is the fact that the first "transistorized" computers, the so-called second generation, such as the IBM 7090/7094, were only then becoming generally available, followed by other machines that began to be used later during this decade, such as the Atlas, the CDC 1604, the IBM 1620, and the IBM 1130. A number of technical developments also occurred at this point. For example, initially in late 1959, the 7090 incorporated an oil-cooled memory system, which was somewhat unreliable, and was replaced by air-cooled memory in 1960. In addition, in 1962-63, IBM introduced the 7040 and 7044 computers, similar to the 7090, but offering somewhat less performance at a much lower price. In the UK, the development of the Atlas introduced so-called "single level storage," which today is usually called "virtual" storage and conceptually permitted the "virtual machine," allowing each programmer to treat the whole of RAM as if available to him or her alone. In short, 1960 essentially marks the point at which mainframe computers began to be both reliable and affordable (for universities and other organizations, that is) and to provide features that supported progressively more interesting software development, leading to greater adoption. For details of the various hardware innovations at this time, in the context of an historical survey, see for instance Rosen [288].

[2] Art Stroud and others also created other programs at Wisconsin during this period, such as RGR, a regression program [124, 323]. These, however, did not introduce new econometric methods, but were interesting for other aspects, as is described later.

[3] Prior to the submission of this thesis, Wickens and Durbin jointly gave a seminar at the LSE to present the theory and results. The program was subsequently used to estimate Rex Bergstrom's model. See also Hausman [144], which this work in part anticipates.

adequate computing facilities. Even in the case of the Brookings Model, a major National Science Foundation-funded project, at Brookings it was only in 1969-70 that a regression program was created that combined an online database with a regression package in such a way as to permit time series to be retrieved, transformed if necessary, and then included in a regression as an integrated process [274, 286]. At that time, almost universally, large card decks, combining both code and data, were fed into often temperamental card reader machines hardwired to the computer in its air conditioned "glass house" [28, 212], although in some cases paper tapes and paper tape readers were used instead, most notably in the UK.

Particularly in retrospect, the impediments are obvious and only slowly dissipated. In the late 1970s, shortly after the birth of the microcomputer, users of mainframe computers normally had access to less than 640K of Random Access Memory (RAM). And throughout those years, more often than not, it continued to be necessary to keypunch the data from printed documents. Whereas by late 1982 or early 1983 the microcomputer potentially provided the individual user a full 640K of RAM, drawing abreast of if not surpassing many mainframes in this respect, it is only much more recently, during the past ten years, that economic data have become widely available in machine-readable form and, in particular, easily downloaded from the Internet. It is only since the mid 1980s that it has been generally possible for someone working alone, with a personal computer and a data base, to construct and solve a large-scale econometric model within a short time period [281].[4] Milton Friedman has described the process of computing regression parameters for a single equation as taking essentially an entire day in the late 1940s [111]. Writing in 1960, Klein provided a similar time estimate, when he said, "Ten or fifteen years ago, the greatest single obstacle to the application of the newly proposed methods of estimation was the heavy computational burden. Single equation methods of correlation analysis had been developed to the point where an individual human computer could cope with practically any problem arising in econometrics. In a matter of weeks, allowing for much experimental calculation, a large system of 10, 20, or even more equations could be estimated by single-equation methods. Many such systems have, in fact, been estimated" [172, p. 868]. Ronald Bodkin [28] more recently describes the early 1960s as the beginning of the time that one could look forward to estimating an equation's parameters in less than hour, taking into account related variable transformations. But the issue is actually not the time of computation: by 1968, it took less than a minute in many if not most cases to make this calculation, but the circumstance that mainframes were then shared facilities meant that it might take as much as 1-8 hours or more for the results to be distributed by the computer operator to the individual user. As a consistent experience, it was not until 1985, or even later, that the *average* user of a computer could expect to run a regression and obtain the results in less than a minute. Today, in contrast, with certain specific exceptions [74], few tasks performed by an econometrician need take more than a few seconds, even the solution of econometric models numbering in the hundreds of equations.

However, this description does not entirely convey the essential characteristic of the microcomputer revolution of the past 25 years, which is the degree of replication it permits. Prior to 1980, mainframe computers were located in many places around the world and some of these could be accessed from a dial-up telephone link from virtually anywhere, providing in principle worldwide use—but in practice comparatively restricted access. The significance of the advent of the personal computer is that for the first time it became commonplace to transfer copies of software, data sets, and even multi-equation models from one computer to another, thus allowing the creation of a true worldwide community of users. As early as 1970, or even before, it was possible to move a model from one machine to another, but not easily; often this involved considerable additional programming and was a rare event. The truth is, other than Klein Model I, very few econometric models were ever mounted on more than a single computer prior to 1984 [281]. The modern ability to share easily both large and small data sets and large and small models across many machines constitutes a major breakthrough, yet one that so far has been only partially realized.

---

[4] The issue here of course is the *general* possibility and the amount of time and skills required. There are a number of examples of econometricians essentially working alone to build and solve large econometric models earlier than this, taking on the roles of model builder, programmer, and data key puncher, among the various activities, but this work was akin to that of the early trans-Appalachian pioneers, alone chopping trees, clearing fields of stones and stumps, and only then planting the first crop. See, for example, the several books by Fair for glimpses of one of these pioneers at work [93,96].

Surprising as it sometimes seems to those who have spent years working with computers, we are still in the early days of econometric computing. The first development of software was essentially an outgrowth of the local needs of particular research projects. During the period prior to 1985, there was in any case little scope to develop for a numerous user population, so that such development as occurred reflected largely own-consumption interests. Sophisticated programs were created that can be regarded as intellectual contributions in their own right, and in particular cases the development of software can be seen as embodying specific econometric methodologies, either implicitly or explicitly [148, chapter 19; 155]. But these activities occurred out-of-sight and out-of-mind of the economics profession generally. It is only in the past few years, with increasingly widespread use of econometric software and the growing appreciation of the inherent difficulties of insuring numeric reliability, that it is slowly becoming evident that there is appropriately a symbiosis between econometric theory and the software that makes this theory operational [148, p. 315; 220, 226, 228, 283]. As early as 1967, it was known among statistical software developers that textbook formulae were not necessarily reliable computationally [206]. Yet only now is it becoming apparent to even specialists, much less the profession generally, that for reliability econometrically sophisticated techniques must be implemented in numerically sophisticated ways. It also may well be necessary for the average user of econometric software to be taught how to use the available software, although in some respects this may be less true of those under the age of 25. In any case, it has been demonstrated repeatedly in recent years that existing packages do not necessarily provide a reliable basis for applied economic research, particularly to the degree the techniques employed are nonlinear in form and users are not careful when applying the software [38, 218-224, 226, 229, 320].

The increasing separation between users and developers of econometric software, as these two roles become more distinct, carries with it a number of implications. In addition to insuring that packages are numerically reliable, there is in particular both a need for this software to be readily useable [52, 283] and the requirement to consider the implications of the way in which econometric techniques are implemented [227]. Some years ago, Kenneth Berk pointed out that although statistical "software should [have] the features to do readily what needs to be done," that in fact there is a "tendency for the user to do what is readily available in the software," that "…packages have enormous influence over…analysis, especially over [that of] the less sophisticated users" [25]. This possibility needs to be much more carefully considered than it has been before. Moreover, there is an increasing need to consider also the relative pedagogic roles now played by textbooks and other printed materials in relation to the existing econometric software and associated manuals. Originally, textbooks consolidated, organized and presented results previously spread throughout the econometrics literature, especially journals. In so doing, they have in the past unquestionably defined current practice, or would have been generally thought to. However, today, it is software that embodies the techniques actually used by economists. All else is purely potential in its effect and to the extent not embodied in software, is likely as the years go by to be increasingly ignored by each new generation that is raised on the microcomputer, not to mention the Playstation, the X-box, and their competitors and successors.

This issue of the *Journal of Economic and Social Measurement* constitutes a first step towards the evaluation of the state of the art of econometrics in terms of the properties of existing software. As indicated, an important aspect of this evaluation is the matter of numeric accuracy, and several of the articles found here address this subject pointedly. However, collectively these articles also raise questions relating to the use of software: implicitly or explicitly, they raise the question whether the software developer's responsibility is limited or unlimited. It is easy enough, in the abstract, to assert that the algorithms that constitute a given program should be numerically sound. However, particularly when nonlinear methods are employed, there are inherent properties of the use process that must be considered. For example, nonlinear problems inherently involve multiple roots, local maxima and minima, and other circumstances that may relate as much to software usability as to strict numeric reliability. There is also the fact that at least two types of packages now exist, at least as polar types: those that offer the user a specific set of techniques, and those that provide him or her essentially a high level econometric programming language, to varying degrees allowing the individual to define the characteristics and properties of the operations performed. A potential pitfall is that, insofar as a package permits the user to determine algorithmically the operations performed, that user takes on the role of a programmer and numeric analyst, shouldering also at least some responsibility to understand the implications of how

particular algorithms should be implemented. Unfortunately, relatively few users appear to realize that they have taken on this burden.

The widespread lack of appreciation of potential problems, so much so that a recent special issue on computing in the *Journal of Economic Perspectives* made absolutely no mention of any aspect of such matters, makes such concerns pertinent in a number of contexts. For example, the *Journal of Economic and Social Measurement* is today rare among those indexed by the *Journal of Economic Literature* to require authors to identify the software used, including at least version and platform, as the basis for published applied work; among economics journals specifically, it is the *only* journal. Moreover, as a matter of apparent editorial policy, other economic journals tend to minimize any discussion of computational issues, as well as economic measurement issues [30]. Does this silence not carry with it the implication that applied results presented in those economic journals are somehow invariant to the software used? To be sure, a number of economics journals have adopted some form of replication policy, mandating the archiving of data [66]. A few require authors to provide information or even code pertaining to self-developed software, but generally restricted to software written in a standard programming language [122]. Particularly in a world in which it remains easy to demonstrate that numeric accuracy and other aspects of the use of software potentially pose serious problems, such policies only barely touch the surface of the solution.

2. The Historical Development of Econometric Software

2.1 The Beginnings

The early development of econometric software can be traced to the first use of the computer by Leontief[5] and Houthakker and colleagues [29, 36, 164, 172, 174, 199-201], the conceptual work by Orcutt [27] and Phillips [197, 198], and the computational work using electromechanical calculators by Tinbergen, Klein, Goldberger, and others [28, 29, 123]. In at least one instance in the early days, there was a close association of the development of the computer and economic research. Richard Stone was among those who then saw that computers could become a powerful research tool, and as a result he appears to have plotted a course accordingly [252, 258, 321].[6] On the one hand, he pursued close links with the Cowles Commission, then at Chicago, and encouraged the Cambridge visits and work of Houthakker, Prais, Orcutt, and others. On the other, because the Cambridge computer, EDSAC, was both the first stored program computer to begin operation (in May 1949) and during the 1950s operated as a general-purpose academic resource, as Director of the Department of Applied Economics (DAE) Stone integrated it into economic research almost immediately.[7] An account of the use of EDSAC 1 to estimate Engel Curves is in Prais and

[5] Leontief's appears to be the first use of a computer by an economist, and therefore deserves pride of place, whether or not his use was otherwise strictly "econometric" in nature. To be precise, Leontief seems to have used the Mark III-IV series of relay/magnetic drum computers at Harvard, which were in part electromechanical, and not actually stored program computers. The development of these was supported by IBM in conjunction with its development of its line of punch card calculating machines. In the interest of full disclosure, from 1980 until his death, Leontief was an Associate Editor and strong supporter of the *Journal of Economic and Social Measurement*.

[6] See, for instance, Stone, Aitchison, and Brown [321], which also outlines a method of programming an expenditure demand model "using a high speed digital machine." The introduction states "It has been adopted as a principle of the research programme of the Department of Applied Economics that the collection and refinement of economic data should go hand in hand with the development of techniques of analysis…we subscribe in general to the belief that a refinement in analytical method requires testing against adequate data before we can decide to proceed further along the same lines or to change the direction of our advance" (p. 1). See also Pesaran and Harcourt [252].

[7] "Beginning operation," as a characteristic of EDSAC, needs to be carefully qualified. In a May 2003 email [310], Lucy Slater writes "In October [1949], when I got to the lab, it consisted of an oscilloscope and a toggle, so that you could enter in binary patterns and see if they produced the correct resulting pattern. By October 1950, it could send a pattern to an old electric typewriter to be checked and read in a

Houthakker [266, 267], with other details in Aitchison and Brown [6]. The subsequently influential paper by Michael Farrell on productive efficiency measurement also included computations based on the use of the EDSAC 1 [101, p. 265, 105, 106]. A JASA paper by Brown, Houthakker, and Prais on "electronic computation in economic statistics" [36] is a particularly informative account of the process of using an early computer (EDSAC), including a description of the context of that work and a comparison of the role of the computer, sorting machines, and the desk calculator to make various types of calculations; this account also provides references to work with the computer by others, as well as references to discussions of the "engineering aspects" of the EDSAC.

The most immediate developmental link, beginning in the mid 1950s, was Lucy Slater, who was both in the DAE, because of Stone, and had helped to establish the programming of EDSAC 1. Working with Farrell, she wrote early regression and matrix algebra packages for the EDSAC 1 and EDSAC 2 [101, p. 265, 313], apparently the first instance of econometric software, also later developing for the DAE a program called REX,[8] a single equation regression program, and GEM, a programming language for matrix calculations [17, 305-308]. Both REX and GEM relied entirely on numeric input, and required the use of specific operator codes; these codes are referenced in the Milestones paper by Slater in this journal issue [313]; aspects of numeric codes and other types of interfaces are discussed in the Stokes paper on software design also in this issue [319]. The EDSAC 1 was of course an exceedingly primitive machine by today's standards.[9] It used mercury delay lines for memory, which was limited initially to 1 kilobyte[10], but the EDSAC 2 in 1957 was a powerhouse: William Peterson [258] reckons that it had at least 32K of memory! And it was powerful enough for Stone to consider using it for a 31 sector Input-Output model; the

---

pattern by a five hole paper tape reader. Much of the hardware was obtained from American Air Force sales of redundant equipment after the war, so it was a case of 'what have we bought today, and what use can we make of it?' Hence the paper tapes. The Lorentz factory in Germany was the source of much of this equipment, which had been used in stock exchanges. [But] by 1951, the first book about EDSAC had been published, 'the Preparation of Programs for an Electronic Digital Computer' by Wilkes, Wheeler, and Gill." See Wilkes et al [346], which describes the capabilities of EDSAC and its characteristics, and the short article by Slater in this journal issue [313]. In his historical survey of electronic computers, Rosen [288] indicates that the first general purpose stored program computer "the EDVAC, was started at the Moore School [of Engineering of the University of Pennsylvania] in 1946. The first to be completed was the EDSAC (Electronic Delay Storage Automatic Calculator) at Cambridge University in England [345]. The EDSAC was started early in 1947 by Professor Maurice Wilkes, who had spent the preceding summer with the computer design group at the University of Pennsylvania, attending the "Moore School Lectures" (http://www.computer50.org/mark1/moore.school/list.html). The EDSAC performed its first computations, the first performed by a stored program computer anywhere, in May 1949. The completion of the EDVAC was delayed by (among other things) the fact that Professors Eckert and Mauchly left the University of Pennsylvania to form their own computer manufacturing company." A 1970 paper by Donald Knuth [180] provides other details on EDVAC, including certain engineering details, and also describes the earliest existant program for a stored program digital computer, written in 1945 by John Von Neumann; Knuth's paper also provides a number of other interesting references.

[8] The earliest independent documentation for this program, but of a later version ported to the Titan (a successor to the EDSAC), is the program manual published in 1967 [314].

[9] In fact, the context of its use required a certain hardiness. Lucy Slater indicates that the machine itself was not in the DAE, but instead "on the top floor of the Anatomy block and was part of the faculty of mathematics." This was near "the physics buildings as Prof. Hartree [who directed day to day operations] was a professor of mathematics and physics, but part of the DAE was in a hut near these buildings so it was easy to get those in the hut interested, long before the rest of the DAE in similar huts at West Road." The "connection with the DAE was a man on a motor bike, Charlie Fienstien who went across from Corn Exchange street, to West Road every morning to fetch us our work and every evening to take back our results. The lady who was my assistant, Ruth Loshak, married him!" [311, 312].

[10] In an earlier version of this paper, I described the EDSAC 1 as initially having 2 kilobytes of memory. However, having read that version, Lucy Slater indicates that, in fact, initially it had just 1 kilobyte, and only later was this enhanced by an extra magnetic tape of other external memory, thus adding a second kilobyte of memory. Engineering details concerning the EDSAC can be found in the book by Wilkes et al [346].

restriction to 31 sectors because a 31 x 31 matrix can just barely fit into 1KB storage. Work on this model was the subject of 12 reports entitled *A Programme for Growth*, the first of which was published in 1962, and some of these reports discuss computational issues. Other materials still exist at the Marshall Library in Cambridge, including a paper by Stone and others about computing .

However, from a worldwide perspective, it is nonetheless true that the development of software for general use by economists began in the mid to late 1960s. At least as a first approximation, it is possible to identify specific centers for this early development, including the Brookings Institution, the aforementioned Cambridge University, the Economic Council of Canada, the London School of Economics, the Massachusetts Institute of Technology, and the Universities of Auckland, Michigan, Pennsylvania, and Wisconsin. With certain exceptions, such as the work done at Auckland by Rex Bergstrom's students [263], [11] at the LSE by James Durbin, David Hendry, Dennis Sargan, Mike Wickens and others [146, 155], at MIT by Robert Hall and others and at Wisconsin by Arnold Zellner, Art Stroud, and colleagues, at the beginning the stimulus for the development of this software was usually to support the estimation and construction of large macroeconometric models. Even in the case of McCracken, it was a move to the Economic Council of Canada in 1965 that provided the incentive to increase the scope of his work. Moreover, particularly in the late 1960s and early 1970s, there was close collaboration between those working on the Brookings Model at many institutions; the Wharton model at the University of Pennsylvania; the MIT-Penn-Federal Reserve model at MIT, the University of Pennsylvania, and the Board of Governors of the Federal Reserve System; and the Bureau of Economic Analysis Model at BEA (formerly the Office of Business Economics) [127, 158] [10, 59, 60] [97, Chapter 1]. A similar web of contacts existed in the UK and between Canadian and US econometrician-developers. These linkages obviously make it difficult to identify the work occurring at any one place as logically preceding that at any other: early individual contributions were made, such as those by Mark Eisner [86, 88], Robert Hall [27], David Hendry [146], Michael McCracken [213], Morris Norman, Ross Preston, George Schink [92, 299], Lucy Slater [306-308, 314, 315], and Arnold Zellner [351, 352], but the lack of a developed historical literature makes it difficult to establish strict precedence in the context of an organized and evolving body of knowledge.

Of course, the development of this software during the 1950s and 1960s took place in an even wider environment than simply econometric applications. Supported by IBM and other such firms and organizations, by the early 1960s documented code libraries were being developed and distributed in a systematic way, providing matrix inversion routines, random number generation, and other code; not always bug free and not always incorporating the best algorithms. In a few cases, even as early as the end of the 1950s, IBM supported the creation of estimation programs by Harry Eisenpress and others [85]. More generally, among econometricians during the middle to late 1960s, code fragments were being casually circulated; for example, subroutine listings from TSP were passed around at Brookings and Wharton in the 1969-70 period, as were also code listings and card decks from BMD and other regression programs that originated in other disciplines. The Census X-11 seasonal adjustment program, both quarterly and monthly versions, became available in 1966-67, programmed by Bakka, Shiskin, and Sommer [302]. In truth, this was a cooperative learning period, during which such ideas as code modularity, exemplified by TSP, and the need for integrated economic data base management, regression, and model solution capabilities were the common currency of thought. A possible metaphor might be largely independent beehives of programming activities, with an occasional interchange of directional messages. It is also important to recognize that the programs of those days were rather rudimentary externally and usually incomplete beyond the specific algorithms to perform a designated operation. Often there were no user guides, other than the source code.[12] And as Longley's seminal paper demonstrated

[11] Which work is described by Phillips and Hall in some detail later in this journal issue [263]. Apparently the work described also involved, to at least some degree, the development of the Bergstrom model, thus was in part model related. However, it was only after the present paper was substantially written that this information, provided by Phillips and Hall, came to my attention, too late to be fully incorporated here.

[12] For example, I still have a 15 page description of the input card deck used to operate a solution program of the late 1960s or early 1970s (there is no date on the document) which is entitled: *THE SIM MODEL SOLUTION PROGRAM by Morris Norman. Description Interpreted from the Fortran by Harold Shapiro With Modifications and Corrections by R.H. Rasche*. Shapiro's name is footnoted "Please send Professor

[206], showing the pitfalls of single precision and the straightfoward implementation of standard textbook formulae, programs then beginning to be used widely were not error free.

The work at the Brookings Institution in the 1960s incorporated several related themes, ranging from the creation of databases by James Craig [79]—based upon programming by Mark Eisner and later partially utilizing the NBER database created by Charlotte Boschan and others [31, 276], to the creation of software to condense Input-Output tables by McCarthy and Renfro [212], to the early development of econometric modeling packages such as MODLER [274, 281, 283] and PLANETS [32, 33]. The first task to which MODLER was applied, and which initially caused its creation, was the estimation of the approximately 200 equation "condensed" Brookings model [115, 116].[13] PLANETS, which became generally available at Brookings in 1971, was used subsequently at Wharton Econometric Forecasting Associates, under the name DAMSEL; it was used until 1987, just before the merger of that organization with Chase Econometric Forecasting Associates. However, in 1966-70, model solution was the paramount cutting edge problem: in addition to somewhat isolated efforts to consider particular aspects of model solutions, such as those reported contemporaneously by Holt [161] and Nagar [239], there was a general effort during that time to determine how best to code, de-bug and solve a large scale model[91, 92, 212]. Early attempts involved even drastic model linearization and heavy reliance on block recursive model structure [162]. Of the first attempts, using the earlier Klein-Goldberger model as a pattern, Charles Holt recalls, "Solving that small model reliably [using the computer] proved to be extremely difficult, so the [initial] solution system [for the Brookings Model] was designed to incorporate five powerful optimization methods including a new algorithm for decomposing the model into its recursive blocks. If we were having trouble solving 20 simultaneous nonlinear equations, [what] about the difficulties that we could face with 300 equations?" [163]. But slowly during this period the method of Gauss-Seidel became generally adopted, based upon recognition by Kuh of the applicability of the method [29, p. 516], first for the Wharton Model [92, 117, 299], leading to the creation of relatively robust model solution programs by the early 1970s.[14] The concept of an online data base as a central element was transmitted to Lexington, Massachusetts in 1968-69, taking root with the creation of the aptly named Data Resources, Inc.

Norman five cents for every successful run. You may bill him an equivalent amount for every failure." Similarly, the brief write ups of the AUTO and ECON regression programs that are among the treasures of my economics library, dated 1968-69, describe the control cards to operate the programs, but also the Fortran subroutines used to read the data from cards, as well as the outline of the subroutine to be used to transform the variables for the regression using that data.

[13] Prior to 1969, the several versions of the Brookings model not only were the "large" version, but as systems of equations also conceptually represented an after-the-fact attempt to merge into a coherent model the independently developed sectors described in the first two Brookings volumes [79,80] and elsewhere [241]. In contrast, the "condensed" model represented at its creation an attempt to evaluate how the degree of aggregation affected the properties of a model, but estimated by Fromm and Renfro at Brookings, in consultation with Klein, Kuh, and others, as a coherent model by initial design, albeit taking into account the sector specifications of the "large" model [113,115]. This model was to a degree subsequently re-estimated in the very early 1970s by Schink during the process of his coding it into a Fortran language simulation package; two variants were created in this process. The need to estimate the condensed version (in those days, at 200 equations, a big model) with a comparatively minimal human input both sparked the initial development of the MODLER software and helped to define the necessary characteristics of an interactive econometric modeling language to support the development and use of large-scale models; extensive conversations with Douglas Bracy and George Schink were also formative.

[14] Since writing this paragraph, I have obtained a manual for "Program Simulate II" [162], apparently written at the University of Wisconsin and published in April 1967. It seems to be the earliest still existing documentation of an econometric model simulation program. It refers both to Newton-Raphson and Gauss-Siedel and generally takes account of the technologies that were then being considered towards solving the problem of solving macroeconometric models. The program the manual describes appears to have had a designed limit of 200 equations and it is not clear to what degree it was ever used in the solution of any actual model. Furthermore, it refers to the assistance of Mark Eisner, identified as then a member of the Brookings Model project, who apparently "helped to establish the IBM compatibility." (p. 8). It is therefore not entirely evident where this work fits into the story.

As described earlier, the Department of Applied Economics (DAE) at Cambridge University was the site of work done by Orcutt [27] in the 1940s, and Farrell, Houthakker [164], Slater, Stone and others in the 1950s. Terry Barker and William Peterson report that in the 1960s there was considerable use of the computer for one-time projects, particularly after the EDSAC 2 was replaced by the Titan in 1965 [17]. Independently, the Titan was also used to implement an early time-sharing system, certainly one of the first [288, p. 31-32]. As a major activity, in 1960, Alan Brown and Richard Stone established the Cambridge Growth Project in the DAE. Stone's Cambridge Growth Model, which initially began in 1962 [322], progressively developed over the next ten to fifteen years into a notably large model that involved a considerable programming effort to both manage its data base and solve it [259]. A characteristic of this model, as it evolved, was its size and complexity. By 1979, for example, the model both numbered approximately 2500 equations and was multi-sectoral, among other things combining aspects of both macroeconometric models and Input-Output models [16]. The dynamically expanding computational problem, seen in terms of the need to solve the joint problems of data base maintenance, numeric computational issues, and presenting results in an organized and intelligible way, was for that time immense. This and other modeling and policy analysis efforts that required computational support at the DAE created a lively environment for software development [11, 51, 257]. Among the results, Slater and Pesaran developed algorithms and programs for parameter estimation [255], out of which ultimately came the present day MicroFit package [253, 254].

The first stage of innovation is proof-of-concept. Subsequently, focus tends to shift from simply doing, however crudely, to doing easily or well. Often, an element of this transition is the relief of tedium: there will always be tediousness associated with some portion of research, but the degree to which this can be minimized can obviously affect the output obtained. The Economic Council of Canada is notable for being the site of the first sustained attempt to define the various aspects of the econometric model building, maintenance, and use process as a computationally integrated activity [213, 214, 216]. Essentially, this was the result of McCracken moving his DATABANK system there and expanding on his earlier work. This system, as its name implies, was designed to manage an economic data base, moreover a substantial one containing as many as 5000 time series or more.[15] The principles of its design therefore attempted to address the multiple storage, retrieval and maintenance problems that such large scale data management implies—even now, and much more so in the mid 1960s, during the fetal stage of large-scale data base management. Programs with evocative names such as MASSAGER [213] and SIMSYS were first created then. As will be described later, subsequently this software, ultimately under the name MOSAIC, was further developed as a coordinated set of procedures intended to provide the support necessary to enable the estimation, construction, and use of large-scale macroeconometric models, including the organization and maintenance of its data base. At the end of the 1960s, this project was only in its early days, but a generation of Canadian economists, including James MacKinnon, are quick to recall their first use of MASSAGER and its siblings.

During the 1960s, the work at the London School of Economics incorporates several distinct contributions. In the earlier 1960s, Durbin, Sargan, Wickens and others used the computer for various purposes including programming estimator algorithms, an example being the effort by Wickens referenced above. This early use, on the one hand, reflected an attempt to make the new technology more easily accessible: to this end, the Statistics Department established general computing support for faculty and graduate students in the form of Ray O'Brien and Wickens, who were given the job of providing the human interface with the University of London's Atlas [342]. On the other hand, achieving technical progress requires a willingness to learn and to innovate: as recounted in a just published article in *Econometric Theory* [152], Hendry describes how "Dennis Sargan hand wired/programmed the computer in the very early 1960s for his 1964 paper. Then he wrote Atlas autocode in the mid 1960s for more `friendly

[15] This project was more ambitious in its aims than it might appear today. For perspective, it should be noted that this work is contemporary with what are now regarded as early attempts by computer scientists to define the principles of data base management [282,301,336]. It is indicative that volume 1 of the *ACM Transactions on Database Systems* was published in 1976, although other ACM publications contain contributions on this topic dating to the 1960s.

programs'!" [150].[16,17] In a similar spirit of innovation, during the mid-1960s, working with Terence Gorman, Bill Phillips, and Sargan, Clifford Wymer created several programs that were subsequently widely used.[18] Hendry himself of course also fell victim to the seductive charms of the machine and his Ph.D. work, begun in the late 1960s, led first to the creation of a software library [146] and subsequently to the creation of a series of programs, culminating in the PcGive and PcFIML packages.[19] In addition to these efforts, the work of Phillips, in the form of the Phillips Economics Computer, is enshrined in the UK Science Museum in South Kensington, London, as a computing milestone[198]. As described by Hendry [152], LSE in the period from 1950 to the 1970s incorporated both distinct points-of-view concerning the precise role of economics, quantitative economics, and econometrics and a practical, hands-on tradition.[20]

This practicality may in fact be more of a marker than might appear at first sight. Worldwide, the period from the late 1940s to the late 1960s was a formative and fertile period for the development of econometric theory [237, 269]. However, two phenomena in particular can be identified with the last ten years of this period and also the 1970s. The first is the initial development of software to actually implement the conceptual constructs of the Cowles studies [171, 181]. The second is the growing disquiet that the specification search process had encountered something of a dead end, roadblocked by both the

---

[16] The EDSAC and the Atlas both were programmed, or could be programmed, using autocode. However, these machines were not directly related. As described earlier, the EDSAC was created at Cambridge University. The Atlas was a second generation machine developed by Ferranti Ltd and the Universities in Manchester [288]. However, there were connections: in a recent email (May 2003), Lucy Slater says, "the language used was Autocode on Edsac1 in the early 1950s and was taken over to Edsac2 which was a bigger version of a similar design, which used ferrite cores instead of valves and so was very much faster. Edsac2 started running in 1958. It was much bigger than Edsac1 as we called the old machine. Whereas Edsac 1 had 512 short 16 bit words at its complete store, Edsac 2 had 2048 32 bit words and was about 10 times faster in its input and output. By this time, most electronic companies had a computer under construction or in the design stage. Ferranti served the Manchester Laboratory, and The Hollorith punched card company, which later became ICL, served our laboratory. J.C.Lyons produced a version of Edsac 2 for commercial sale, as did Elliots and English Electric. The Ferranti company which had made its name during the war, with radar and similar devices, now produced a commercial machine called Mercury, which was followed by Pegasus. And then the Atlas series, by 1962. The Atlas … in London was one of these. Its autocode was very similar to the old code which had been used in Cambridge and Manchester, with of course the needed additions to deal with magnetic tape readers, and enlarged backing store devices. The third machine in Cambridge was Titan, which started work in 1965 and was an upgraded version of an Atlas."

[17] Other such recollections exist. Sargan customarily lectured without notes. Referring to this, Tim Harrison remembered recently an occasion that, as a student, he was in Sargan's office when the phone rang—someone from the Atlas computer center reporting a problem—whereupon Sargan quickly dictated new code over the phone. There is no guarantee that the program then immediately worked, but it may have.

[18] These include Resimul, developed in 1967/68 as a Full Information Maximum Likelihood estimator of coefficients consisting of nonlinear functions of parameters, initially using a modified version of Durbin's Newton-Raphsen procedure; Continest, developed in 1968 to calculate eigenvalues and their asymptotic standard errors; and Predic, developed at the same time to calculate asymptotic standard errors of dynamic or static forecasts. The programs were subsequently used in Australasia, Europe, UK, and the US. For further details, see the entry for WYSEA in the software compendium at the end of this issue and the references found there.

[19] The development of these more recently has of course been transferred to Nuffield College, Oxford [74,153]. The package called here PcGive was called PC-Give up to version 6.x and PcGive from version 7 onwards. PcGive has been adopted here as the name. A similar nominal transformation occurred in the case of PcFIML.

[20] As a case in point, M.Sc. students choosing the mathematical economics option in the late 1960s not only battled their way through such things as the Kuhn-Tucker saddlepoint and Radner turnpike theorems, not to mention those associated with separating hyperplanes, but were also called upon (in an exam setting) to compute numerically a reasonably detailed Simplex solution, presumably in order to demonstrate a mastery of the associated computational process.

problem of non-nested hypotheses and other aspects and assumptions associated with the implementation of the "traditional" approach [67]. However, in contrast to the angst expressed by Leamer [196] and the nihilism by Sims [304],[21] at the LSE the development of the methodology of specification search became a challenge, rooted in the Sargan paper of 1964 [65, 152, 293], but also the countervailing influences of Friedrich von Hayek, Karl Popper, A.W.H. Phillips, Lionel Robbins, and the Staff Seminar on Methodology, Measurement and Testing (M^2T), among others. Throughout the 1960s and subsequently, there was at the LSE the closely related development of software.

Somewhat earlier, the first use of the computer with econometric models apparently occurred at the University of Michigan in conjunction with the estimation of the Klein-Goldberger Model[172].[22] Initially, this model was estimated and solved using desk top calculators, applying the methods described by Goldberger later in this journal issue [123]. But by 1954, as recounted by Klein, computation "was partially shifted to electronic computers for the estimation of moments in preparation for parameter estimation" [29, p. 515; 174]. A similar methodology was subsequently used at the Oxford Institute of Statistics for the Klein-Ball-Hazelwood-Vandome model [175]. Solution of the Klein-Goldberger model posed a problem that was resolved in part by simplification. The model was linearized, as described by Suits [324, chapter 2], preparatory to solving it using a Monroe electromechanical desk calculator, employing "an iterative procedure that [Klein] had devised" [325]. Experience with the solution of this model provided a starting point for considering the solution of the Wharton and Brookings models, as mentioned earlier. However, later 1950s and early 1960s Michigan models were also linearized and then solved by matrix inversion. Subsequently, in the early to mid 1960s an attempt was apparently made to create a general solution program for the Michigan models of that time on an early IBM mainframe, a program called Obake, Japanese for "ghost," but the details are now phantasmic [29, p. 515; 325].

At MIT, prior to the installation of the IBM 1620 and the 407 card reader, Marchant desk calculators were to be found in the basement of Sloan. David Belsley recalls that "even a small project could tie up a couple of people for several days just producing the input for the normal equations," not to mention the mistakes that could be made, Franklin Fisher stopping by from time to time for a tally [22]. Robert Hall began the development of TSP in 1966 on the 1620 [140] and continued this development at Berkeley on a CDC 6400 and a remote Univac 1108. Associated with Hall in this program development were Ray Fair, Robert Gordon, Charles Bischoff and Richard Sutch.[23] Under the general direction of Edwin Kuh, a series of programs were subsequently developed at MIT, including GREMLIN [18] and TROLL [86, 88, 193]. Hall recalls learning from Mark Eisner "how to do symbolic differentiation," as well as developing matrix routines for TROLL, and subsequently developing "nonlinear estimation procedures using analytical differentiation, in the time and frequency domains," as a bi-coastal research assistant for Kuh and Dale Jorgenson both in Cambridge and at Berkeley [141]. For some years, MIT was preeminently the site of a considerable amount of work on computer algorithms [62-64, 159, 188, 189], as well as work on regression diagnostics that was closely related to this algorithm development [19, 20, 24]. Kuh was not then the only person who understood the promise of the electronic computer, as we have seen, but one of his significant contributions was his willingness to act on his belief that econometric computation was and is in principle an intellectual activity on a par with the development of theoretical econometrics. Belsley recalls that the ethos of the Center "directly reflected the strong interdisciplinary nature of Ed Kuh. Ed had an amazing ability to see when two researchers doing seemingly unrelated things were actually doing the same sort of thing" [22] .

---

[21] T.C. Liu of course earlier made a somewhat similar argument to that put forward by Sims, concerning the identifiability of macroeconometric models [204]; however, see also Liu's qualification in the 1975 Brookings volume [205].

[22] However, see in addition Phillips and Hall [263], who indicate that Bergstrom, at the University of Cambridge, was also an early user of the computer to support parameter estimation during roughly the same time period.

[23] Later, at Princeton, Ray Fair and J. Phillip Cooper contributed to the development of TSP [138]. Fair remembers that "at MIT, I programmed TSCORC and TSHILU into TSP, which was two stage least squares with first order autoregressive errors" [96,98].

At the Economic Research Unit (ERU) at the University of Pennsylvania, even in the 1970s desktop calculators could still be found—apparently by then disused, although there was a rumor that in the dead of night Lawrence Klein might still be observed cranking one from time to time, just to verify that the IBM 360 was producing the right numbers. But at nearly all hours in the early to mid 1960s, the ERU and its "spin-off" Wharton Econometric Forecasting Associates, under the direction of Klein and Gerry Adams, was the site of innovative work by Morris Norman, Ross Preston and George Schink that led to the development not only of parameter estimation packages, such as Norman's AUTO and ECON, based upon initial programming by Preston [2, 245], but also early work on the development of model solution algorithms [34, 128, 246, 299]. Both the Brookings and Wharton models were solved there in the mid-to-late 1960s initially using the Newton method [92, Chapter IV; 212, especially footnote 9], which was later replaced by the Gauss-Seidel method, being less cumbersome; the programs to do this were created by George Schink [299] working with Vijaya Duggal, Michael McCarthy and Ross Preston[24]. A substantial number of other, more supplementary, programming projects were also undertaken in connection with the various large and small model building efforts then under way at Penn [10, 112], not to mention a variety of more generalized empirical investigations [183-185]. One such was the updating of the Wharton Indices of Capacity Utilization, programmed by Renfro in 1972 under the direction of Robert Summers [3, 176]. This project was an extension of the earlier work by Klein and Summers, programmed by Ross Preston and others in 1966 [177, 178]. A difference was that in 1972 this work was intended to provide software that could be used generally; surprisingly, the resulting program continued to be used at Wharton EFA for at least the next 15 years. Although not then demanding computationally, and thus not in a league with the creation of solution programs, the significance of this type of support programming is that it nonetheless made possible the non-tedious generation of useful statistics. Fifteen to twenty years prior to that, the handling of calculations to determine relative peaks and other characteristics of the raw data would have been truly burdensome [29, p. 510-514].

However, in order to appreciate these developments fully, then and subsequently, it is necessary to understand certain aspects of the context of the 1960s and later mainframe computing environment. Throughout its first 40 to 50 years, the mainframe computer was hot, literally. It generated a substantial amount of heat that needed to be dissipated, reflecting initially the use of vacuum tubes, but more generally the heat energy from the close proximity of many electrical components, large and individually heat-producing—large particularly in comparison with today. Always, the computational speed of the electronic computer has depended upon minimizing the distance between components, and hence even today heat generation remains a critical issue. But in the early days, which certainly included the 1960s, computers were housed in a (normally) glass enclosed, air conditioned environment.[25] Within that environment, the machine's operator(s) tended the machine, monitoring its operation, rebooting it whenever necessary, mounting and dismounting tapes (and later removable hard disks), and sometimes performing such other tasks as distributing any printed output—which in those days was the intelligible result of the operations performed. Sometimes, other, subordinate operators would be responsible for distributing the printed output of the machine, which as the years wore on increasingly took place at remote

[24] In his 1992 *Econometric Theory* article, McCarthy provides a slightly confusing report about this effort, indicating that the Newton method was not used, because it was too formidable computationally: "No one in the 1960s (or the early 1970s) had any intention of using Newton methods, though they were cited in [the Cowles Commission] Monograph 10; the computational burden simply seemed too formidable." However, he immediately thereafter states that the method used was that reported in Evans-Klein [92], which was first programmed by Schink, as just described, and was in fact a Newton method [212]. The contradiction is more apparent than real, for McCarthy later indicated in private conversation that when he wrote this article he was thinking of a simultaneous solution of the full model, which was never attempted at this time.

[25] Normally, but not invariably. Lucy Slater recalls that originally, "in the room where the first Edsac lived, there was no air conditioning, not even a fan, but we had a door to the roof outside and several large windows which opened wide. In summer the temperature was frequently up to 100. In the room which held Edsac II there were several large fans which could be turned on and off as well as several large windows to open. The room which held the Titan was however, completely enclosed and air conditioned. There the main problem was the fast paper tape reader, which was very sensitive to ladies' face powder, or a flying insect!" [309].

sites, that is in other (usually nearby) buildings, although by the early 1980s such printers might be located even 500 (or more) miles away. But, particularly in the early years, only the operator had a live video monitor and the output of that was rather esoteric. The revolutionary aspect of the first networked machines was that individual users did have their own screens (or paper-based terminals with keyboards), displaying their own typed input and results, but even in this case the operator of the central machine needed to be contacted to mount tapes and perform other critical operations. Of course, at this time, such networked machines were rare.

More characteristically, in those days, particularly when IBM machines were used, programs were input in the form of a sequence of punched cards (collectively called card decks) via a machine known, logically enough, as a card reader. The cards, each three and one quarter inches by seven and a three-eights in dimension, were conceptually subdivided into 80 columns and 12 rows, ten of which were numbered, with rectangular holes punched in particular columns, achieved using a somewhat typewriter-like keyboarded machine called a keypunch. The absence of any holes in a particular column would be interpreted by the computer as a blank character, or a zero, as appropriate. In many cases, more than a single hole would be punched in a column. These card decks would consist of "job control" cards that indicated to the machine the name (and usually account number) of the user and characteristics of the machine resources required by that "job", followed by cards that consisted of the user's program, followed by the data used by that program.[26] As the years passed, the program might be stored on a tape, or even a removable hard disk unit, and the job control cards would then provide instructions to the machine

[26] The RGR program, written by Arthur Stroud at the University of Wisconsin at the beginning of the 1960s offers an interesting example of data input [124, 323]. The data were organized on the punch cards "by observation": one after another, for each given observation in time, data on each variable was punched onto a card, organized in fixed column fields, with that time period's observation on each variable thus being punched onto each card progressively, so that the first field referred to the first variable, the second to the second, and so on. The effect was, as the cards were read, that the observations were then located in a matrix, held in memory, itself organized so that the columns were variables and the rows observations. It was apparently possible to then select a subset of the observations (by choosing the beginning and ending rows) and a subset of the variables, if wished, in order to control the subsequent computation of the sums of squares and cross products (raw moments) matrix from which then the particular variables used in the regression were in turn selected so as to form the matrices and vectors directly used to compute parameter estimates. This process of forming the relevant matrices is not so very different to that used even today, although inasmuch as the cards consisted of only 80 columns, there was immediately a problem concerning how to read in data sets that might contain more than 8-16 variables (assuming that each number required a field of 10 to 5 spaces). However, this was a relatively minor problem that could be solved by permitting 2 or more cards per observation as the data deck was read in. Much more troublesome was explaining to the program user exactly how to control these (unseen) operations in order to select both dependent and regressor variables. As a consequence, the user's guide not only explains to the user the process of forming the matrices, but provides forms to fill out that corresponded to the control cards that actually directed these operations. The user is told [124]:

> "From your point of view this means that you must give the data deck to the programmer and inform him as to the following:
>
> a) Where the variables are on the data deck.
> a) For each correlation problem: which variables are to be selected, which observations are to be selected, and which output is to be printed.
> a) For each regression problem: under which correlation problem it is to be subsumed, which variables are to be used and in what order, which (if any) intermediate regressions you want, and which output is to be printed. You will also include various identifying information.
>
> The programmer will translate your information into a form intelligible to the program and computer. A standard format for presenting your information to the programmer has been developed."

At times during this process, a ram may have been sacrificed, or various incantations recited.

operator, indicating the tape(s) or hard disk unit to mount. In this case, the card deck would contain just job control cards and the data for the program, although this configuration was much more common in the 1970s than in the 1960s. Referring to the early 1970s, Mike Young recently recalled the days "when the entire database and code for the FRB-Penn-MIT model existed in a complete form in two boxes of cards which lived in the trunk of my 1958 Plymouth Fury. It never occurred to me that an accident might well have destroyed about 10 man years worth of research" [347]. By the end of the 1970s, the card deck, if used, would be likely to contain only cards directing how the computer should perform the "job": the data too would be stored on tape or hard disk. This innovation, had it occurred earlier, would have saved Michael McCarthy the dismay of watching his card boxes actually being run over by a delivery truck in the early 1970s [212, p. 388].

Self contained jobs would be run as just described. However, it was also possible even in the 1960s to store the data on tapes. In 1966-68, econometric software used with large scale econometric models at Brookings and elsewhere took the form of separate but related programs: for instance, one for data base maintenance, one to make transformations, and one to perform regressions. The data base was normally stored on a tape and, in order to manage it, the maintenance program addressed the tape and added or revised individual series observations, in the process writing a new tape. The transformations program extracted series from the revised data base on this second tape, performed transformations and wrote these to a third tape. This third tape was then read by the regression package and the regressions were performed, the results printed out somewhat cryptically using a "line printer," the object referred to earlier as producing the printed output, so-called because it printed (at Brookings) a 132 character line essentially simultaneously—inasmuch as there was ordinarily one line printer per machine at that time, output speed was important.[27] The printed output was cryptic not so much because of any characteristic of the line-printer, which was capable of printing a perfectly human-intelligible document (although not with the range of typefaces the laser printer is capable of), but rather because in those days the programmer and consumer were usually one and the same. Only once others began to use the software did it become important to begin to format the output so that it would be intelligible to others. Programmers, aka young economists performing the role of research assistants, would be separately responsible for writing out the results by hand, in standard written notation, so that others could see what results had been obtained.

The card decks themselves could be printed out, using the computer and the line printer, so as to provide a listing of the operations performed. However, this listing tended to be difficult to decipher. For example, individual time series were usually identified in this context by number, that number normally being the record number of the variable on the tape. When transformations were performed, the type of transformation would also be identified by a number, so that a representative transformation instruction might take the form:

42 22 3 16

having the interpretation that the observations on the variable in record 22 should be multiplied (operation 3) by the observations on the variable in record 16 and the result stored in record number 42. It is immediately evident that a deck of 400 cards, each having such an instruction punched into it, was both difficult to interpret after the fact (even by the person who originally created it) and likely to contain at least

---

[27] In fact, one of the important aspects of the historical development of the electronic computer that is often overlooked has been the lagging development of peripheral devices. As noted in passing by Rosen [288], early computers did not print output directly, but instead often either punched cards or in some cases produced tapes, which then were used to produce printed output. For instance, the UNIVAC I had a particularly advanced buffered magnetic tape system, but initially incorporated a device "the Uniprinter, which operated at electric typewriter speed directly from magnetic tape" (p. 11). "The early scientific computers were designed in accordance with a philosophy that assumed that scientific computing was characterized by little or no input or output. The [IBM] 701 and early installations of the 704 used an on-line card reader (150 cards per minute) for input and printed output could be obtained only from an on-line printer that could print 150 short lines or 75 full lines per minute" (p. 15). It was only in the 1960s that input and output began to be efficient, and that only by the standards of those days.

one error, often a number of errors. Modern econometric software packages permit the user to write an instruction like:

$$GDP = C + I + G + X - M$$

but such an instruction must be interpreted by the program, usually converting it into a set of machine instructions telling the computer to retrieve observations from a sequence of record locations, perform specific operations, and then to store the result in a particular record. Anyone familiar with a spreadsheet program will immediately realize that even today some vestiges of the 1960s environment remain, inasmuch as such instructions still refer to rows and columns, even if the operators are specified as +, -, * , /. On the other hand, only an economist is likely to understand the GDP identity and be able instantly to identify its components.

In the late 1960s, the computational challenge for those who would subsequently develop today's econometric software can be characterized as follows. First, there was the most basic computational challenge: algorithms, which is to say the sequence of operations necessary to perform some recognizable set of mathematical/statistical calculations. These had to be created so as to enable anyone to use the computer to generate results corresponding to those described in some journal article or textbook or defined by the user. This development of algorithms was begun in the 1960s, but only begun. It has not been completed yet, both because econometrics is a still developing discipline and because this is difficult work not all the implications of which are yet understood. Second, there was the problem of marshalling the resources of the computer, so as to permit a sequence of operations to be performed in a coordinated way. In the case of substantial projects particularly, data bases obviously need to be organized and maintained. Even to perform a minimal amount of work, the data need to be managed and transformed. For example, the estimation of parameters usually involves some transformation of the original data. Finally, the user of the computer, particularly if he or she did not wish to spend a substantial amount of time learning to program, needed a way to communicate with the machine easily in order to tell it what operations should be performed, where to find the data to use, and how to display the results, to name only certain of the things that have to be done.

Characteristically then, and also even today, programs to support macroeconometric models were usually developed with the stress placed upon the harnessing of computer resources and with a focus on making these programs generally usable—in those days ordinarily reflecting the need to support a team of economists. Programs developed as a response to an economist's personal needs tend to be much less developed in terms of data management support and the human interface, with the stress placed upon such things as algorithms associated with particular types of parameter estimators. The initial orientation is often important as a potential, if implicit design restriction: subsequently, it is usually relatively easy to add a new estimation method to an existing program, but it can be difficult to modify a program originally developed for a specialized purpose to support use in a larger-scale context or use that involves a high degree of process integration—in much the same sense that once a building has been constructed it is normally feasible to modernize particular rooms, but much more difficult to increase the number and purpose of rooms without affecting the usability of other existing rooms.

Of course, hardware was an issue, and the experience at the DAE in Cambridge, although not necessarily precisely typical, is illustrative. As mentioned, the Titan replaced EDSAC 2 in 1965, and was finally to gain a Fortran compiler in 1970, but the machine was designed to be programmed in Autocode, a proprietary high-level language. Autocode provided the capability to do serious econometrics, but restricted the ability of the software to circulate outside Cambridge. Interestingly, although the speed and core memory (RAM, that is) appears limited in retrospect, at the time the principal noticeable constraint was not either of these attributes, but instead the storage space—hard disk in later years. Peterson reports that "when I started on the Growth Project, the allocation for an 8-person project was only 300K bytes" [258]. Similarly, across the Atlantic, until 1970 at Brookings the machine was an IBM 7040, which provided tape storage, but no hard disks, and core memory hardly greater than the maximum the *original* IBM PC permitted to be installed on its motherboard, namely 64KB. Word size variations between machines, and perhaps faulty human memory, makes specific magnitude comparisons difficult. But the idea is not too misleading: certainly there was no one, anywhere, with any machine containing a 512MB

RAM chip and a 60 GB hard drive, much less a notebook. In fact, across the Pacific, working on the IBM 1130 in the late 1960s, Peter Phillips had to contend with the limitation that "the machine had only 7K core storage and so the program had to be written in segments that were linked together using Call Link statements in Fortran, the data from one stage being written to disk and then read back from disk when the next stage of the program was loaded. On one occasion, the line printer jammed and in order not to lose my operator's license I had to spend the rest of the night reading the technical manual in order to learn how to dismantle the line printer, unravel, clean and rewind the 80 feet of printer ribbon so that the machine was operational when the day shift arrived at 8:00am" [262]. Evidently, at the end of the 1960s, it was generally necessary to be a "programmer," if not always an actual licensed computer operator, in order to use a computer; this changed, to a degree, in the 1970s.

2.2 The Promise of the 1970s

The 1970s and into the early 1980s are now remembered for Stagflation and related problems, but computationally, it was a formative period. In Canada and the United States, the establishment of firms such as Data Resources, Chase Econometrics, and Wharton Econometric Forecasting Associates and the development of macroeconometric models in governmental organizations, such as the Economic Council of Canada and the U.S. Department of Commerce, led to the creation of sizeable data bases and dial-up mainframe time-sharing use of software to access those data bases; this was also the period during which government agencies in Canada and the United States began in earnest to develop data distribution initiatives that today have migrated to the Internet [264, 276, 282, 328]. In Europe, but most specifically in the UK, the Cambridge Growth Project, the London Business School and National Institute of Economic and Social Research played similar roles to the economic forecasting firms in the United States. The Treasury Model and that of the Bank of England represented the public sector. In addition, a number of individual projects were undertaken that have since led to important computational developments on both sides of the Atlantic, including work done by Fair at Princeton and Yale; Hendry and colleagues at the London School of Economics; and Barker, Pesaran, Peterson, Slater, and colleagues at Cambridge University.

One of the difficulties in assessing the development of econometric software even during this period is that of determining the extent of original, independent research. Not only has important work often been undocumented in the standard economics literature, but as indicated earlier a considerable amount of technology transfer also occurred. The result is that it is hard to determine who first did what when, for it is generally true that econometric models require local programming support, or at least did in the "old days," whether or not independently inspired software development took place. A comprehensive history of the period would take into account, in Canada and the US, the support work done at places such as Indiana University, UCLA, and other university and organization-based macroeconometric model projects [89]. Software development was also required in order to support the development of microsimulation models [248, 290], as well as INFORUM at the University of Maryland [8]. In Europe, there have obviously been computational projects associated with econometric models in a number of countries [29]. In various other parts of the world, there have also been a number of software development projects, some of which are described for the first time in this journal issue [261, 263]. The degree to which the present account is comprehensive as regards past innovative work will only be evident in the years to come, assuming of course better reporting.

As mentioned earlier, particularly during the 1970s, the Center for Computational Reseach in Economics and Management Science at MIT, under the direction of Edwin Kuh and Roy Welsch, was an important center for econometric software development that both directly and indirectly sponsored a number of subsequent initiatives. The flagship system was TROLL [43, 193, 348], begun by Eisner in 1968 and which became operational in 1972 [86, 88].[28] During the 1970s there were a number of new

[28] The earliest software development work on a prototype system, TROLL/0, apparently began in 1966. However, as Eisner announced in 1972 [86], "In the fall of 1968 work was begun on a completely redesigned and much expanded system. This system TROLL/1, is now programmed and operates on the IBM 360/67."

facilities incorporated into it, such as GREMLIN, containing David Belsley's nonlinear estimation algorithms, and the "Regression Diagnostics" of Belsley, Kuh, and Welsch [18, 23, 24]. However, the Center's work was far more generally based: in the early 1970s, a cooperative association with the National Bureau for Economic Research [187], then led by John Meyer, resulted, among other things, in the publication of the journal *Annals of Economic and Social Measurement*, which during its six or seven year lifetime published a series of important articles on software and data base development and some of the seminal work on economic control theory applications that would later lead to the founding of a second journal, the *Journal of Economic Dynamics and Control*, as well as to the creation in the 1990s of the Society for Computational Economics, largely at the instigation of Hans Amman and David Kendrick [170] and the journal *Computational Economics*.

The three major econometric forecasting firms operating initially in the United States, Data Resources, Chase Econometrics, and Wharton Econometric Forecasting Associates, provide interesting contrasts in the way they each developed software for their own and clients' use. Data Resources Inc. (DRI) was founded in 1968 by Otto Eckstein, Gary Fromm, and others. It distinctively promoted the concept of providing large scale data bases to a wide market, but essentially business economists, the restriction reflecting the cost of the services provided. The contemporaneous development of mainframe-based dial-up and dedicated-line telecommunications capabilities meant that, beginning in approximately 1970, it became possible for DRI to offer online time sharing computing services, mixing software and a substantial time series data base that grew to literally millions of time series [276]. To offer these services required the development of a supporting software system. DRI's first such system, started in 1968, combined an adaptation of TSP by Robert Hall, called EPL [141] with a separate model solution program. The second, under the name EPS, was developed during the 1970s, principally by Robert Lacey, but with important design contributions by Hall and Edward Green [55, 56]. These systems, developing sequentially over time, provided DRI clients with the capability to perform regressions, display tables and graphs, and, by 1980, even solve some models, although in fact the model solution facilities were never a fully integrated feature, particularly to the degree of being able to handle DRI's larger models as a matter of course [29, 167, 271].

Chase Econometric Associates, founded in 1970 by Michael Evans with the backing of the Chase Manhattan Bank, was also strongly oriented towards the business economics community and slowly developed similar online data and software support for its clients. In response to the development of EPS by DRI, Chase contracted with Mark Eisner and others in the late 1970s to create XSIM as a commercial adaptation of TROLL. XSIM incorporated a number of the matrix routines originally developed by Robert Hall for TROLL as well as the symbolic differentiation routines developed by Eisner and Hall[88, 141]. XSIM, which reached full flower during the period 1980-1983, was a much more integrated econometric modeling package than EPS, and considerably more user friendly than TROLL [83, 102]. Although Chase Econometrics was essentially an economic consulting and forecasting firm, an associated entity, Interactive Data Corporation, provided a broad range of computer technology-related services and products, a number of which were created using the macro language and facilities of XSIM, although other econometric packages were also offered, such as AUTOBJ, now known as Autobox [285].

Wharton Econometric Forecasting Associates (Wharton EFA), until about 1980 a non-profit corporation wholly owned by the University of Pennsylvania, was somewhat less successful in the development of state-of-the-art online systems during the late 1970s and early 1980s. However, beginning in the very early 1970s, Wharton EFA was the innovator in the development of software and models embedded in software that were made available on 9-track computer tapes to its clients [34, 128]. The clients included not only business economists but also research centers at a number of universities [179]. The programs developed by Wharton EFA were generally less sophisticated in the range of parameter estimation facilities, but in the early 1970s more developed in their model solution capabilities, which influenced work at other US locales, including MIT [211, 246]. The DAMSEL system, which as noted earlier was essentially a re-badged PLANETS, was developed in the mid to late 1970s by Douglas Bracy and others at Wharton EFA to provide parameter estimation facilities in an online database context, but never attained the degree of integrated processing of either EPS or XSIM later in that decade.

At the DAE in Cambridge, the 1970s were years of fruitful activity. The Titan was replaced by an IBM 370 in 1972, which among other things permitted programs to be written in Fortran; so did the Titan, but only from 1970. Most importantly, packages driven by a series of numeric codes were replaced by those with more modern interfaces. Peterson developed IDIOM (Internal Dynamic Input-Output Model) and MREG (Matrix and REGression package), both with command line interfaces[259]. These were batch programs, reflecting the IBM's limited interactive capabilities then and there. Peterson feels that "a dedicated FORTRAN program was [then] the only way to write a program which was sufficiently fast and compact" to support the development of the Cambridge Growth Model, which already was beginning to bulk up in size, dwarfing most other econometric models [258]. The quantity of data associated with this model, as well as its size, posed a succession of software design challenges, ranging from the need to efficiently maintain and update a substantial number of data series, to the need to provide immediate access to the data for estimation and simulation[17, 257, 260].

More generally in the UK in the 1970s, there were a number of computational initiatives: a model solution system called NIMODEL was developed at the National Institute of Economic and Social Research (NIESR), principally by Richard Pierse. This package was somewhat integrated with a data base system but without parameter estimation capabilities, and in modernized form is still available today [142, 326]. At Imperial College, University of London, Berc Rustem, John Prescot and Sean Holly developed control algorithms for large econometric models [160]. At the Treasury, David Ramsden and others also developed model solution software. The Klein-Ball-Vandome model, created at the Oxford Institute of Statistics, was transformed into the London Business School model by Ball and others. Models were created also at Liverpool and Southampton and elsewhere in the UK [13, 329]. Essentially, during this period, the existence of any working large-scale econometric model implied also the existence of programming support, but as mentioned earlier the developmental linkages are not well documented in most cases, although the work at Southampton in particular was finally to cross the Atlantic in the early 1980s and blossom in Philadelphia in 1985 as AREMOS.

Meanwhile, at the LSE, the development of software that began as a software library called AUTOREG [146] increasingly became the handmaiden of what has emerged since as the LSE methodology. As Hendry recounts, "initially the programs focused on 'optimal estimation' based on the implicit assumption that dynamic models under study were more or less correctly specified. The tests offered were mainly tests of model specification, where the null was just a special case of an already estimated more general model. Then an increasing number of diagnostic tests was included for misspecification…gradually leading to the implementation of 'model building' procedures." [148, p. 314] Hendry's development of AUTOREG thus shines forth as software with a purpose, possibly in contrast to other software development, which can be seen as often undertaken simply to perform a specific task of the moment. On the other hand, it is also possible to observe, in considering other software systems, that when (metaphorically speaking) bullets are flying, large tasks are performed without introspection that may nonetheless later appear to have led to a coherent and well-defined result. When comparing AUTOREG to other systems, there is an interesting question whether these others should be regarded as embodying a contrasting philosophy or a neutral concept?

Towards the end of the 1970s, AUTOREG evolved into David Hendry's Give. In the process, this program began to affect other programming efforts in UK, reflecting the growing influence there of the LSE General to Specific methodology. At the National Institute, REG-X was created by Steven Hall and others as direct clone of Give [142]. At that time, Give was a mainframe program. REG-X as initially developed was a Fortran-based program that ran on a Tandy microcomputer and produced essentially the same output structure as Give. When the National Institute purchased a mini-computer, Hall ported REG-X to that, where it ran from card image files and was extensively used locally. Whether this reflects a quintessential English desire to hold on to the past or not, REG-X is possibly the only example of an econometric software package ported as a replacement from a microcomputer to an older class of machine.

In Canada, at the Economic Council of Canada and later at Informetrica, there was further development of MASSAGER, SIMSYS, and other software to support the creation and solution of macroeconometric models, particularly the CANDIDE and TIM models, the first during 1971-79 and the second beginning in 1979. From the 1960s, MASSAGER was also used by other Canadian government

agencies, including the Bank of Canada and the National Energy Board. Finally, MASSAGER "73" was released into the public domain. During this time, SIMSYS emerged as a package for model solution and "equation maintenance." A distinguishing feature was general support for model maintenance and solution, with the equations initially specified in an algebraic format that was translated by the system into Fortran code, which then became an integral component of the solution software. At Informetrica, founded by Michael McCracken, the MOSAIC system incorporating such features was developed subsequently, becoming operational in or about 1979 [166]. MOSAIC was designed as a group of distinct but related subsystems or procedures controlled by a common mathematical-textual language [215]. The subsystems consisted of program control, data management, basic arithmetic, regression analysis, and Forecasting and Seasonal Adjustment. Organized in a somewhat similar manner as SAS, MOSAIC's program control module served as a command interpreting interface between the user and the several applications subsystems.

During this period, there were also individual software development efforts that are best viewed independently of any particular organizational context. Ray Fair's work is an example: this work spans the development of software for both parameter estimation and particular types of model solution and its philosophy is today embodied in the FP (Fair-Parke) program [96, 99, 100]. Characteristically, during the 1970s, Fair's models were re-estimated frequently, using specific estimation techniques, and were solved without the use of constant adjustments [93, 94, 96]; this particular orientation is reflected in the properties and progressive development of the associated software. In contrast, TSP continued to develop as a broadly-based package, inherently general in its implied philosophy [77, 139], as did also MODLER [78, 281]. However, in the case of TSP, the focus was specifically estimation and within this context relatively comprehensive in scope. TSP's nonlinear and system estimation algorithms, in particular, were affected by results presented at the MIT Center for Computational Research: Bronwyn Hall recalls as fundamental to the development of TSP an important conference on nonlinear estimation methods held at the MIT center, a number of the papers from which were published in the October 1974 issue of the *Annals of Economic and Social Measurement* and elsewhere [138, 270]. MODLER's development during the 1970s represented a third way, stressing instead the broader aspects of the process of model construction and use as an integrated activity. Neither as specifically methodological as Fair's approach, nor quite so general in its estimation facilities as TSP, MODLER began from the concept that ideally the computer could provide the ability to interpret mathematical-textual instructions as if written on a piece of paper, with the fundamental difference that in a computer context these would instead be operational. This orientation reflected the interactive context of MODLER's early development: the PDP-10 at the end of the 1960s, which provided the individual user with a remote terminal that included a keyboard and CRT screen, making this concept immediately appealing as a goal. To realize this goal required the development of a natural language interpreter and the solution of a number of process integration problems in order to permit an isomorphic representation of the model code on the screen and in machine language [281, 286].

However, not everyone focused quite so directly on equation systems. Packages that began to be developed during the 1970s include Autobox, B34S, BRAP, LIMDEP, RATS, and SHAZAM. In common with many other programs, each of these appears to have been created initially either in reaction to the absence of other, appropriate software, or to solve a particular problem. LIMDEP began in 1974 at the University of Wisconsin as an implementation of Nerlove and Press's multinormal model. Later it also implemented Ray Fair's EM algorithm for the tobit model [95].[29] It has finally become "an integrated program for the estimation and analysis of linear and nonlinear single equation models, using any of three types of data: cross section, time series and panel data" [134]. The forerunner of RATS was SPECTRE, written by Christopher Sims and others. This package was developed "to overcome some limitations of existing software that affected Sims' research in the 1970's, by providing spectral analysis and also the ability to run long unrestricted distributed lags." [70]. Tom Doan added ARIMA and VAR capabilities and

[29] This characterization is Bill Greene's. In a recent email, Marc Nerlove [243] has pointed out that, particularly in the statistics literature, it is generally accepted that the origin of the EM algorithm is Dempster et al [61] and Rubin [289]. However, in response, Bill Greene noted that, in his 1977 *Econometrica* paper, Fair produced, apparently independently, an algorithm that is equivalent in its characteristics to the EM method, but quite specific to the tobit context, and that this is the method used in LIMDEP [133].

subsequently looping and other programming features and changed its name to RATS. As a mainframe package, RATS was written in Fortran, but was re-written in C in 1991, after it had already been converted for use on the PC. The earliest development of SHAZAM traces to Kenneth White's undergraduate days at Northwestern, but the program's main development began at Wisconsin in the early1970s on a UNIVAC 1108. It survived transplantation to the IBM 370 at Rice, Michigan and the University of British Columbia [340, 341]. Autobox was also begun at Wisconsin, being an extension of the original "Pack Program" created by David Pack under the direction of George Box in the late 1960s; among other things, David Reilly added a "a rule-based heuristics system that is today the Autobox approach to ARIMA modeling." Design aspects of the development of B34S, among other packages, are considered by Stokes elsewhere in this issue [319], but this program was begun at the University of Chicago essentially to meet local needs and has since developed into a relatively comprehensive estimation package.

Considering these individual initiatives collectively, it is possible to highlight several salient areas of their development. On the one hand, econometricians began to incorporate into their packages both additional parameter estimation techniques and more sophisticated methods that involved nonlinear algorithms [18]. Linear-in-parameter estimation algorithms generally center on the inversion of a matrix, in the simplest case the symmetric matrix of sums of squares and cross products, and the best algorithms achieve both the inverse and, simultaneously, the parameter estimates. Nonlinear estimation involves a number of further considerations, including the need to compute derivative values accurately, usually in the context of a Hessian matrix. Nonlinear estimators, including Full Information Maximum Likelihood, provided a challenge that was then hard for econometric developers to resist, although nonlinear estimators were not commonly used in applied work until perhaps recently, particularly with the development of GARCH and other special application techniques [227, 270]. Meanwhile, in the case of software being developed within macroeconometic model projects, the focus tended to be on the integration of large data bases with estimation and display capabilities, and in a few cases, also the integration of model solution components, reflecting both that not everything can be done at once and that OLS and other linear methods were those most commonly used for large scale econometric models. Thus in the 1970s, the still continuing pattern was firmly set of some developers choosing to incorporate mathematically sophisticated techniques, and others developing packages involving different but similarly computationally challenging features such as program control via natural language commands—involving the need to parse and evaluate inputs and translate conventional mathematical expressions, as written, to polish notation and other more computationally tractable representations—as well as implement large-scale data base management, tighter process integration, and other such features.

However, as the established programs in the 1970s became progressively more feature-rich in their particular ways, other packages began to be created that reflected econometricians' desire to perform specific operations not included in the readily available packages. In certain cases, as with SHAZAM, the original stimulus for development was that packages readily to hand such as BMD, developed out of the Biometric tradition, were not especially suitable to econometric applications specifically. In other cases, as with LIMDEP (which is obviously a play on LIMited DEPendent variable), packages were developed as extensions within the econometric environment. Context is important: starting in the 1970s, panel data began to become more prevalent, with the development of longitudinal data bases associated with the National Longitudinal Surveys, the Panel Survey of Income Dynamics, and more recently the Survey of Income and Program Participation in the United States, or the Household Panel Survey, the National Child Development Survey, or the Social Change and Economic Life Initiative in the United Kingdom [210, 242, 282, 327]. These are of course simply examples, and, ultimately, a number of public use microdata samples have become available, drawn from tax records and other administrative data collection programs as well as surveys intentionally undertaken to support research [39, 232, 273].

Simultaneously, there were important extensions in coverage. For example, in 1972, Arnold Zellner and a number of colleagues began to develop BRAP and other programs, in order to provide a solid computational foundation for Bayesian estimation and analysis [1, 349]. This case is an obvious example of an instance in which existing packages may well not cater to the interests and needs of a specific group of econometricians. One issue that this type of software development raises is whether there is a need, in such cases, to allow users to add facilities to existing programs? In the absence of this capability, anyone with a minority interest must respond by creating not only algorithms to perform a particular task, but in

fact a complete program. Yet there is no difference necessarily between the Bayesian and other approaches as regards such things as data base maintenance facilities and many of the other ancillary program features. In the 1970s, it was sometimes possible of course to obtain source code from an existing program, and simply to embed a particular technique in it, but this process still required programming skills. However, need is not always a bad thing: especially in the early days, when program features were not well developed, the creation of additional programs was in itself socially beneficial, inasmuch as it created a cadre of people interested, however initially grudgingly, in software development.

These general comments introduce the embedding concept and identify possible applications, but the development of BRAP is also worth examining in slightly more detail in order to reveal more specifically what might be involved in the process of embedding a particular set of techniques as a component of an existing program. The earliest version of BRAP was in fact embedded in B34S,[30] then an existing program with the necessary facilities to load and mange data, having a structure very much like TSP at that time, combined with a user interface that could display results [318]. In this context, the data loading element of B34S essentially functioned as a preprocessor to prepare the data for use by the BRAP add-in. It then executed BRAP. Finally B34S used its facilities to produce for the user the output from BRAP. As explained in the Stokes paper on software design in this journal issue [319], this type of embedding is relatively easily accomplished in the context of a procedure-driven program, such as B34S. Nevertheless, although this arrangement formed only a temporary way station in BRAP's development, it is still interesting as a example of both developmental relationships between programs and a relatively early attempt to minimize program development time in order to produce a result. In a Windows context, an essentially similar development has occurred recently with the development of GiveWin as a front end interface to not only PcGive, but also the current version of TSP.

The econometric software developments of the 1970s taken as a whole, even if they did not necessarily become fully realized then, do reflect that those years—particularly the early years—were a time of creative expansion. And although not always in a consciously coordinated way—in fact more often than not just the opposite—the effect of the various individual efforts in multiple countries and many contexts has ultimately resulted in a remarkable, jointly-developed software edifice. There was both a broadening and a deepening in the range of econometric techniques. Successive editions of econometric textbooks over the past 40 years, such as those of Johnston in 1963, 1972, 1984 and Johnston-DiNardo in 1997, as well as the generational development of textbooks by students of earlier textbook authors, obviously demonstrate changes in emphasis in the estimators described, but they also map both a broadening and deepening of the subject coverage. And to a degree, simply the existence of the computer undoubtedly facilitated this expansion. Thus, for example, in 1970, Edward Prescott, at Brookings temporarily from the University of Pennsylvania, was then in the early stages of the formulation of his method of stochastic parameter estimation and was interested to discover how to embody this concept in software [48] [268]. Although only a relatively small subset of all estimators described in the econometrics literature have been included in econometric software packages generally, it has been common since the 1960s for econometricians to use the computer when developing estimators and methodologies, as for example in the cases of Durbin, Sargan, Wickens, and Zellner mentioned earlier. Moreover, particularly in the case of econometric software packages developed out of and for academic use, much of the software development has reflected the perceived individual needs of the econometricians doing the development.

However, lest any inference be drawn that the decade of the 1970s was manifestly a golden age, it needs also to be said that econometric software development then was distinctly a minority sport. For one thing, it is important to realize just how much smaller the number of prospective users was then as compared to now, and to recognize that the computer was not ubiquitous, nor even particularly intensively used by economists. Not every economist spent the weekend in the keypunch room; indeed, for most

[30] In the interest of expositional simplicity, this description involves a degree of poetic license in the use here of the word "earliest." To be absolutely precise, in 1967, in its original incarnation, BRAP was created as an integral part of B34T, the predecessor of B34S. In 1972, Zeller and others developed a standalone BRAP, in the process separating its code from (then) B34S. It was this earliest *standalone* version of BRAP that was operationally embedded as here described, permitting the programs to be operated jointly, but without any further actual intermingling of code.

economists, the pencil was the most important applied research tool. Moreover, there were certain market conditions that limited prospects. Econometric software is the focus of this account and so far the focal point has been the supply side. But it is also pertinent to consider this software in its role as statistical software, a broader classification that includes also SAS and SPSS. In the 1970s and before, the market for such software was very much affected by the fact that computers then tended to be organizationally owned and managed, making it generally desirable that any software acquired should be broadly applicable. Regression, as a statistical technique, fits this requirement. But it is also true that in these years, particularly the early years, even among economists the demand was often for regression as a technique, rather than for an econometric software package per se. In the mainframe and even minicomputer applications software market, statistical software that included regression capabilities proved a natural contender, provided that it also included a wide range of other statistical techniques and options. Packages such as SAS and SPSS became widely used in both universities and elsewhere, even in comparison to other categorical types of software.

Reflecting the commercial imperative, SAS and SPSS, among others, were developed to satisfy the apparent needs of a broad range of users. However, inasmuch as these users were not homogenous in their interests and requirements, in time such packages progressively began to be developed to meet the perceived demands of identifiable categories; for example, the SAS/ETS (SAS/Economic Time Series) module was created at the end of the 1970s and included both frankly econometric techniques and an econometric model solution facility [7, 291, 296-298]. It was possible to buy the SAS/ETS user guide in many, if not most university bookstores, particularly in those instances that SAS/ETS was mounted on one or more machines there. In a few cases, econometric software packages became reasonably widely used, and TSP is an example of this. However, Hendry speaks both for himself and others when he said that "most econometric software has embodied an implicit methodology and modeling approach, but widely distributed programs have usually lagged some years behind the state-of-the-art technical econometric frontier" [148, p. 315]; particularly in certain cases, this lag spurred local software development. General statistical packages may have been the more marketable, but the need to serve a broad market also meant that actively developing market segments were not necessarily well served.

During this time, there was also development for additional computing "platforms," including the minicomputer. IBM mainframes were pervasive, at least in the United States, but during the 1970s the Digital Equipment Corporation began to develop its VAX minicomputers, following the PDP series of time sharing machines that began in the 1960s. The minicomputer provided a somewhat more intimate environment for software development, being conceived as a machine for a smaller group of users, a "department" rather than the "enterprise." In an academic context, the university might be served by one or more mainframes, but the economics department might be better served by one or more minicomputers. It is not clear that the minicomputer per se noticeably influenced the character of econometric software development to the degree that this machine type deserves to be considered separately; however there is little doubt that it made computing more widely available and, at the very least, permitted development to begin that might not have otherwise, either at that time or perhaps even ultimately. The econometrician who begins to program at the age of 50 or later is unusual, to say the least; it is far more common to begin younger. Therefore access to a machine (or not) is generally a critical element in the career of econometricians who also program. Another aspect of the computing environment at this time, which also bears on the issue of access to a computer, was the fact that programs did not always cross national boundaries.

Even within countries multiple frustrations existed. In the UK, for example, IBM computers were not so common at university computing centers as in the US; ICL and CDC machines were, particularly ICL. As a consequence, TSP and other packages from across the pond were not readily available, leading to the local development of programs such as HASH and, later, DEMOS, both developed by Tim Harrison, then at the University of Southampton. However, remote job entry (RJE) terminals that linked provincial universities to the University of London computer center, or to the University of Cambridge, did exist and provided some degree of cooperative development nationally. If the focus of this paper were impediments to the development of econometric software during the 1970s, quite a lot could be said about the balkanization effects created worldwide by incompatibilities among operating systems and among machines from different manufacturers. For example, it could be difficult using an IBM machine to read a

tape created by a CDC machine, or a tape from an IBM using an ICL. In addition, everywhere, the resource constraints were binding: some computing centers made it difficult to obtain turnaround in less than 4 hours for jobs that required even 128 K of RAM; in almost any institutional setting crossing the 512K boundary, or sometimes an earlier trip-point, could mean a 12 or even 24 hours wait. Even into 1980s, such problems persisted.

2.3 The Last Days of Empire: Mainframe Developments in the 1980s

In general, it is difficult to summarize the state of play at the beginning of the 1980s. The year 1980, or any near year, is not a natural dividing line. The first generally available, albeit build-it-yourself microcomputer, the Altair 8800, was introduced in January 1975, but as a type the microcomputer did not of course supplant other machines until well into the 1980s. As with any new development, even one that captures widespread attention, the adoption process takes time, particularly in those instances in which the implement, which is to say the hardware, is not in itself ready to use, and not only evolves rapidly for the next 20 years and more, but requires the development of a complementary product, the software, in order to be used effectively and widely. In or about 1980, the Apple microcomputer was already well known and as a somewhat experimental object was being adopted: Apples could be found adjacent to mainframe computers, usually in a separate room. In headquarter office settings, they sometimes could be found: in Texas, for example, the King Ranch was making productive use of an Apple in 1982-83. But even in the case of the IBM PC, it took 3-4 years from introduction before it began to be adopted widely in an everyday context. Just as in 1915 the horse was still commonly used in transportation, during at least the first half of the 1980s, the mainframe computer continued to be used at least as widely as before, if not more so. Actually, although the term "supplant" has been used, for ease of exposition, in fact the microcomputer did not supplant the mainframe so much as become a tool widely used by people most of whom had never before used a computer. There was a change in the compositional mix of mainframe applications, reflecting the migration of particular types of applications to the PC. And certainly, for econometric applications, the microcomputer moved analytic processing out of the glass house and onto the desktop, but even in this case the vast majority of people who began to use it had never before operated a keypunch nor loaded a card reader.

Francis Diebold has recently asserted that "Structural Keynesian macroeconomic forecasting, based on postulated systems of decision rules, enjoyed a golden age in the 1950s and 1960s following the advances in Keynesian theory in the 1930s and 1940s" [69]. It is not entirely obvious how to interpret this statement, either in context or out of it, but if it is a statement about the use of structural macroeconometric models, the dates are obviously wrong. Notwithstanding the problems encountered incorporating stagflation into these models beginning in the mid 1970s, the golden age of this type of model (if it has yet occurred) is much more appropriately identified with the years from 1975 to 1985 or later—measured in terms of either money spent to purchase access to forecasts made using them or the number of models active. Until 1985, almost all this activity took place in the context of mainframe econometric software, even if in 1985 there was a rush to implement microcomputer software in order to construct and solve such models.

One of the implications of the popularity of forecasts during this period may be that, had there been no revolutionary development, caused by the microcomputer in the 1980s and the Internet in the 1990s, it is likely that for mainframe econometric software the early 1980s might stand out as the beginning of a period of consolidation and refinement, much as the Edwardian era appears in the aftermath of World War I had there been no war. Or perhaps, given this revolution, such an interpretation is actually quite appropriate. There is good information on the state of the art of econometric software at the beginning of the 1980s. As well as in individual articles and books [23, 244, 255, 281], it is contained in such publications as the 1980 supplementary volume *Computation in Econometric Models* of the *Journal of Econometrics* and the special issue on modeling languages in the 1983 *Journal of Economic Dynamics and Control* that documented the June 1981 annual meeting of the Society for Economic Dynamics and Control in Copenhagen[Dent, 1980 #1438;Kendrick, 1983 #1435;Control, 1981 #1437].

In North America, at the beginning of the 1980s, integrated mainframe econometric modeling systems such as EPS, MODLER, and XSIM were organized as online data and software systems in the context of wide area telecommunications-linked networks [276]. In addition, in Canada, Statistics Canada's CANSIM provided a substantial online data base of Canadian data. For the US, although not yet online, the Bureau of Labor Statistics LABSTAT system had been developed and contained disaggregated time series data on employment, prices, and other related statistics [233, 234, 278]. As yet, there were organizational details to be worked out concerning the mechanisms for the distribution of US government produced data versus the "data vendors" such as DRI, Chase Econometrics, Haver Analytics, Wharton EFA and others[42, 46, 81]. However, even at the regional level there were systems, such as those in Indiana and Kentucky, that provided dialup and networked access to data resources and computing facilities by academic, government, and other users [4, 5, 277, 280, 282].

In Europe, the proliferation of macroeconometric models [29] had resulted in the establishment of at least one organization in almost all countries that supported the development of models and software, as well as, in some cases, some form of online data base system, such as that produced in Calabria by Pitagora SpA, a subsidiary of the San Paulo bank. Not all these organizations were in the process of developing extensive telecommunications-linked networks. But DRI, Chase, and Wharton EFA each had offices in a number of countries, and in many countries, particularly in Europe, there was at least one indigenous firm. And of course at universities and governmental organizations, models were operating. A number of these models were participants in the LINK project, which resulted in the distribution and development of software on a fairly broad scale [12], although not always in the independent development of software. In particular countries, somewhat similar cooperative efforts were established in order to foster technology transfers as a subordinate goal: in 1983, at Warwick University, the ESRC Macroeconomic Modelling Bureau was founded to "to improve the accessibility of macroeconomic models of the UK economy, to promote general understanding of the properties of these models, and to undertake its own comparative and methodological research" [332-335], including a distribution mechanism for model solution software and facilities. Quite naturally, the focus at Warwick at the outset was mainframe computing facilities.

In the UK, as has been discussed, there were in particular a number of economic model-based forecasting and policy analysis organizations that developed during the post-1960 period, especially during the 1970s and 1980s. Those that were notably prominent in the development of software for model estimation and solution were the Cambridge Growth Project and the National Institute of Economic and Social Research, as indicated earlier. Both in Cambridge and at the NIESR software development continued into the 1980s. In each case, this development reflected the size and the characteristics of the embedded models, as well as the particular way in which these models were expected to be used. During the first half of the 1980s, Steven Hall created a new model solution package CEF, that was an extension of Richard Pierse's NIMODEL. CEF was created in addition to REG-X, mentioned earlier. The CEF software has continued to be developed at the NIESR and in microcomputer form is today used with the Institute's UK and World models, the latter called NIGEM. This software is tailored to these models particularly, and approximately 30 central banks around the world subscribe to NIGEM embedded in it. However, if software persists, scholars are peripatetic. Progressively developed by Hall, CEF and REG-X moved with him to the Bank of England in the late 1980s and to the London Business School Centre for Economic Forecasting in the early 1990s, where the DataView component of MODLER was used to provide the data base management support—which combination may illustrate several aspects of the technology transfer process. Sean Holly, similarly, appears to have traveled from London to Sheffield to Cambridge with code. "Have code, will travel" might well be the motto of many econometricians of the second half of the 20th century, and by no means only in the UK.

Today, the thought of anyone setting up a buggy whip factory in 1910 seems whimsical. So too, setting up mainframe-based facilities in the early 1980s already appears in a somewhat similar light. However, this interpretation is entirely a matter of hindsight. From the perspective of 1981 or even 1983, the microcomputer appeared still toy-like to many observers; in late 1983, many organizations kept PCs in separate rooms as shared facilities and an organization that employed 1000 or more people might have but a single machine. A famous article in *Datamation* magazine in 1983 was one entitled "Real programmers don't use Pascal," a panegyric on the joys of mainframe Fortran and its manly properties[265]. At the 1981 American Economic Association annual meeting, Otto Eckstein, commenting as the discussant for a paper

that in part described early work on the use of the IBM PC [279], characterized the microcomputer as needing at least 5 years of development before it could be considered seriously. Even with hindsight, the Eckstein comments are perfectly justifiable as a prediction of the time when the microcomputer was likely to become generally used for analytic work in preference to the mainframe—although these comments also possibly help to explain the change in DRI from a computing leader in the 1970s to a distinct follower in the next 20 years. It is important to be ahead of the curve during revolutions.

The image of British exports to the United States in the 1960s and 1970s centers on the Beatles, the Rolling Stones, and other, even more Hardrock groups. However, in the 1980s there were software exports as well. As mentioned earlier, in the late 1970s, Tim Harrison began the development of DEMOS (Dynamic Econometric MOdelling System) at the University of Southampton to support the modeling effort there. Subsequently, his work came to be supported by the French Atomic Energy agency, CiSi. In 1980, CiSi purchased Wharton Econometric Forecasting Associates, one of the aims being to marry an extension of the DEMOS package with the Wharton models. Later renamed AREMOS, this software was developed first as a mainframe package, with a considerable emphasis on data base management. The mainframe version has since continued to be developed, but during 1984-85, a microcomputer version was also developed, very much in conjunction with the mainframe version, but also to a degree influenced by the properties of MODLER, the microcomputer version of which was then being used by Wharton EFA and a number of its clients. AREMOS, in its early development, was an interesting system particularly because of the close integration of the mainframe and microcomputer versions, enabling the user of the PC version to extract data series from the mainframe in a relatively effortless way. When Chase Econometrics was merged with Wharton EFA in 1987, AREMOS became the software of choice of the new entity, the WEFA Group, and XSIM fell by the wayside, although for internal company use it was not quite such a walkover as this may sound. During the past year, with the merger of the WEFA Group and what was originally DRI, to form Global Insight, AREMOS is in the process of apparently superceding the now rather antiquated EPS.

The development of MODLER as a mainframe system until 1981 and a microcomputer system since is described in considerable detail elsewhere [281, 283]. However, briefly, the mainframe version of this software was made available during the second half of the 1970s and the early 1980s on two wide area networks. One of these was maintained by the Kentucky state government, and was used by its employees and, to some degree, the Council of State Governments and other national organizations. The other formed the Kentucky Educational Computing Network (KECNET) and was used mainly by state-supported universities and related organizations throughout Kentucky. MODLER was implemented in the context of a slightly more general software system, known as the Online Retrieval and Computational Language for Economists (ORACLE) that was an extension of the 1973 version of MODLER. The data base contained a substantial amount of national, state, and local area time series data [276, 277], including approximately 14,000 monthly, quarterly, and annual time series for the state and its subdivisions[275]. The overall system was known as the Kentucky Economic Information System (KEIS) and from the beginning of 1976 its purpose was to provide data and software facilities, including cartographic displays, to both academic and governmental users, but also the public at large at various specific sites and via telephone links [4, 5, 275, 287]. In that context, MODLER was used to estimate, build, and solve a variety of econometric models, mainly models of the state of Kentucky and certain of its component geographic areas. MODLER ceased to operate in this context towards the end of 1981, but the KEIS continued to operate as a mainframe data distribution system until the middle of the 1990s.

These online data base and analytic processing systems tended as a group to be less well known among academic economists. Historically, there are three reasons for this. The first is that they either cost too much for widespread academic use, with the fees for software often included in subscriptions for model forecasts and charges for access to data and consulting, or else to be local to their particular environment. The second is that they characteristically did not attempt to provide either leading-edge estimation capabilities or solution facilities for models that incorporated rational expectations and other academically interesting solution features. The context in which these systems operated tended to restrict development to those features likely to be used immediately and intensively by a broad group of people. Third, as software, with the exception of MODLER, from early 1983, and AREMOS, from 1985-86, these systems were available during the 1980s only as mainframe-based time sharing systems, using either dial-up

telecommunications or much more expensive T-1 lines, or else as custom installations on corporate and other institutional mainframes; in the case of XSIM and EPS, these custom installations could cost hundreds of thousands of dollars. Paradoxically, the average user of these systems, particularly when operated as mainframe systems, often used them to perform simple OLS regressions, if that, or to display graphs, charts, or tables, not for particularly sophisticated work. The use to which the systems have been put, being focused on light analytics, has meant that since the late 1980s both AREMOS and EPS in the PC environment have been placed at a competitive price disadvantage to Excel and other spreadsheet packages and, since the early 1990s, also to EViews.

From an academic perspective, it is easy to dismiss the type of system exemplified by AREMOS, EPS, or XSIM. Being "commercial," and responding to their user base first and foremost, these packages have always tended to offer less sophisticated econometric facilities than the "most advanced" academically rooted systems, systems such as MicroFit, PcGive or TSP, among the general systems, or other more particular and more specialist systems, such as those developed by Fair. However, that said, it is also true that when typical users are taken into account, the offerings of AREMOS, EPS, and XSIM in the 1970s and 1980s, like the more ubiquitous Excel or even Word in the 1990s, are possibly more representative of the econometric facilities actually *used* by economists. Day-to-day average use does not involve as sharp a difference between the world of the business and the academic economist as might be thought from a theoretical literature review. Of course, the migration of the LSE methodology across the Atlantic has resulted in some changes, but best practice and actual use are not the same thing—in truth, it is not hard to overestimate the comfort of the average user, academic or otherwise, with cutting edge technologies. It also needs to be taken into account that during the 1970s and 1980s, in fact until the second half of the 1990s, most econometric software packages, of whatever classification, offered a relatively unsophisticated range of supplementary statistics [283]; the distinctions between such systems, in terms of parameter estimation facilities, centered almost wholly on the range of estimation methods offered. Moreover, it is still true that in the contest for greatest use, FIML still finds OLS hard to beat.

An additional interesting case is the development of software used in conjunction with the staff research work for the Board of Governors of the Federal Reserve System.[31] In the early 1970s and throughout much of that decade, the software used to support the development of the Federal Reserve-MIT-Penn model was obtained from the Economic Research Unit at the University of Pennsylvania or its affiliated company, Wharton EFA, or was created by extending the features of that software, as well as those of other well-known software packages. However, at the end of the 1970s, a software system called Modeleasy began to be developed as an extension of a existing non-econometric system, Speakeasy [45], originally created by Stanley Cohen, then a theoretical physicist at the Argonne National Laboratory. Modeleasy incorporates the Speakeasy system and in the early 1980s was "defined as the Speakeasy processor enhanced with an FRB econometric vocabulary called FEDEASY" [47]. The development of this system subsequently has been the work of James Condie, then at the Board of Governors, Andrea Cividini and Giuseppe Bruno at the Bank of Italy, and William Teeters at the Econometric Modeling & Computing Corporation, although it also incorporates some work of Alfred Norman and others [47, p. 75].

Of course, as indicated, it is likely that the mainframe software system most used by economists during the 1980s was SAS. The mainframe version of SAS offered not only its standard regression facilities, but also the ETS supplement, as mentioned earlier, which permitted the creation and use of small macroeconometric models. For the estimation of parameters, and general statistical work, SAS was undoubtedly the most used single system. For the creation and use of large macroeconometric models, there is very little evidence of the use of this system, reflecting that during all the past 50 years relatively few large scale econometric models have been created independently. The 1983 article by Drud [78],

[31] In keeping with the overall focus of this paper on econometric software as defined earlier, the emphasis here is on the modeling software at the Board of Governors. However, other, quite closely related software has been developed at the Fed, including for example W.P. Cleveland's weekly seasonal adjustment program, in part based upon earlier programming work done by David Pack and Mike Grupe at the University of Wisconsin. Ideally, such work should also be considered in the present context, but time and space considerations place limits. However, this omission should not be inferred to imply a qualitative judgment; only a definitional restriction.

describing the individual model solution capabilities of then existing software systems, also provides essentially a census of the mainframe econometric model support software systems at the start of the 1980s, but not all the systems described there have ever been used with any known model.

2.4 The Advent of the Microcomputer

The microcomputer has its origins in the early to mid 1970s, but insofar as econometric software is concerned, the critical development was the introduction in August 1981 of the IBM Personal Computer. At its introduction, the original IBM PC offered only 16KB of RAM, with the option to add a further 48 KB (at a price of at least $100), for a total of 64KB. This machine also offered diskette storage in the form of 160KB single sided diskettes. The early programming languages available included Assembler, Interpreted Basic, Fortran, and Pascal. By design, floating point operations were supported in hardware by a separate supplementary FPU chip, the 8087, which adopted the IEEE standard and offered the possibility of performing numerically accurate computations not only faster than using software arithmetic emulation, but with a native level of precision beyond that of contemporaneous mainframes.[32] The primary CPU, the Intel 8088, was a hybrid, internally a 16 bit chip, but externally 8 bit, running at a blinding 4.77 MHz. Today, these specifications are exceeded by that of most PDAs and possibly even some digital cameras, but at the time the introduction of the PC was exhilarating.

In addition to having been introduced by IBM, which provided an aura of seriousness, the most significant feature of the IBM PC was its open architecture. Constructed of essentially off-the-shelf parts, the machine could be enhanced by its users. Consequently, within 6 months, third party manufacturers were offering memory boards that expanded the available RAM to as much as 320K, then 540 Kb (limited by a Bios quirk), and shortly thereafter to 640Kb. In 1982 IBM began to offer dual-sided diskette drives that provided 320Kb of removable storage; with the introduction of DOS 2.0 in 1983, the capacity of these immediately increased to 360 Kb. But not until March of 1983, with the introduction of the IBM PC XT with its 10 megabyte hard drive [40, 250], did the PC become sufficiently advanced to support truly "large-scale" computing. Even then it was the introduction of the PC AT in September 1984, with DOS 3.0, a 6 MHz CPU, a 1.2 Kb diskette drive, and a 20 MB hard drive, that set the stage for subsequent hardware developments [41]. However, in 1984 a PC with dual 360 Kb diskette drives, an 8087 FPU, and 640 Kb of RAM did provide the capability to permit the solution of the PC-Mark7/MODLER 250+ equation Wharton Quarterly Econometric Model for 12 quarters in less than 4 minutes [281]. And by October 1984, the 1200 baud (1.2 Kb) modem was a reality, even if not present on many desks at that time [303]. Of course, this brief account ignores the creation of a variety of clones and nearly PC-compatible machines during the 1981 to 1984 time period that, together with the PC, actually launched the revolution, but descriptions of

[32] The 8087 FPU chip performed operations with a single instruction that to perform with the 8088 alone might require many. The taking of logs and exponentiation, in particular, were therefore speeded up substantially. Some idea of the difference can be gained from knowing that the multiplication of two two-byte numbers would take 32 microseconds using the 8088 alone, whereas the 8087 could perform this operation in 27 microseconds. However, this is a best case for the 8088, which was a 16 bit chip (16 bits equals two bytes and 16 bits can hold only numbers that fall within the range –32768 to 32767). The 8087 in contrast contained 80 bit registers, obviously giving it a leg up as the necessary bit length of numbers increased. Moreover, the standard arithmetic operations (add, subtract, multiply, and divide), as well as the calculation of square roots, tangents, arctangents, logs, and base 2 exponents on numbers up to 18 digits in length were built into the hardware. In addition, various constants including zero, one, π, and standard log representations are included in the 8087's instruction sets. Consequently when such things as exponentiation or the taking of logs are considered, the 8087 was as much as 100 times faster than the 8088. Today's Pentium III and Pentium IV CPUs incorporate floating point operations and thus it is no longer necessary to purchase and install (or have installed) a separate FPU chip. Further details on the 8087 are provided in Startz [317] and in a useful article by Sarnak and Jaffe in the *PCTech journal* [295]. Another view of the 8087 is provided by Stephen Fried [110]. The adoption of the IEEE standard was important because it established a trans-platform standard for floating point computations, among other things allowing the PC to be adopted as a scientific tool.

these are readily available elsewhere.

Actually, the first attempt to perform a regression on the microcomputer predates the IBM PC. In fact there were apparently at least three early attempts: at the end of the 1970s, prior to his better known work on Lotus 1-2-3, Mitch Kapor, then at MIT, created a microcomputer regression program for the Apple II, loosely based on TROLL, which he called Tiny Troll [87].[33] At about that time, in the UK, Steven Hall implemented REG-X on a Tandy microcomputer [142], as described earlier. In 1981, in California, David Lilien implemented MicroTSP on the Apple II using interpreted Basic. For Lilien, in particular, the attraction of this specific machine was its bitmapped screen and high resolution color graphics. He recalls, "because I wanted MicroTSP to support graphics and to have a full screen interactive interface I chose the Apple II as my development platform" [202]. However, despite these efforts, once the IBM PC was introduced, in short order it became the platform of choice for econometric software. Not only did it quickly define its market, but among other things the Apple did not offer numeric processing capabilities sufficient to make this alternative more than mildly attractive to developers of computationally intensive software. Notwithstanding that more than a decade was to pass before the Windows machine in the 1990s became even marginally visually competitive with earlier Apple computers, for a variety of well-known reasons the IBM and compatibles became generally market dominant even in the early 1980s. Consequently, before long, as had Kapor, MicroTSP migrated to the PC, becoming a compiled Basic program in 1984, at which point support for its Apple II version ceased [202]. In general, although there are modern examples of econometric software packages being developed for the Apple, including—since 1990—a version of TSP, for the past 22 years the PC has been where the main action is.

In the beginning, the transition of econometric software packages from mainframe to microcomputer was nevertheless relatively slow to occur. Ivor Francis' 1981 compendium of mainframe statistical software packages [108] included several econometric programs, among them B34S, MODLER (in the context of KEIS/ORACLE), SHAZAM, TROLL, and TSP (2 separate versions). Of these packages, only MODLER is present in the Elsevier microcomputer software catalog of Fall 1983,[34] produced from the International Software Database [68], although both listings also include the more general statistical programs SAS and SPSS. These compilations are not necessarily comprehensive, so that exclusion of any particular program is not indicative per se, but rather simply illustrative of the speed of development of microcomputer programs and their conversion from the mainframe environment. Early microcomputer compilers for Fortran, Pascal and other compiled programming languages were notoriously difficult to work with, in contrast to their mainframe equivalents. Furthermore, recall that it was not until early 1985 that most organizations began to buy PCs in any quantity.

However, by the end of 1985, econometric software programs began to proliferate. By then, AREMOS, AutoBox (then called AutoBJ), PcGive, RATS, Shazam, and Soritec were operating on both the

[33] Kapor at this point became the product manager for VisiCalc and aparently took the plotting facilities from Tiny Troll to provide the basis for VisiPlot. Tiny Troll itself then evolved into VisiTrend. Having sold the rights to these products to VisiCorp, Kapor left and then began the development of Lotus 1-2-3. See http://www.crn.com/sections/special/supplement/763/763p63_hof.asp for additional details.

[34] The fact that a significant portion of MODLER's initial software design occurred in the context of the IBM 7040 and the DEC PDP-10, both 1960s era machines with comparatively minimal RAM and other computational resources, was of considerable benefit 11 years later when MODLER was ported to the original IBM PC. Its consequently parsimonious use of resources, not to mention its implementation on a variety of IBM 360/370 and later machines under several different operating systems, may help to explain why this complex and comprehensive program was one of the first previously mainframe-resident programs to be successfully ported to the microcomputer with all its facilities essentially intact. MODLER was (and still is) capable of performing all its operations, including solving simultaneous models of as many as 1000 equations in less than 460K RAM, without overlays, although the Windows version, making use of a Visual Basic front end as a user interface, now takes up much more RAM including that interface. Both overlays and the need to maximize RAM efficiency were a feature of life in the case of the IBM 7040 and the PDP-10. Of possible incidental interest, Bill Gates and Paul Allen wrote the first version (dialect) of BASIC for a personal computer using an 8080 emulator running on a PDP-10.

mainframe and the PC. Gauss was generally available on the PC and, as mentioned, MicroTSP had been converted from the Apple. By the later 1980s, with the introduction of microcomputer compilers that permitted a direct port of mainframe programs, a number of additional econometric packages were converted to the PC, or else directly developed there, including such still-existing packages as LimDep, MicroFit, REG-X, Stata, and TSP, as well as packages such as EAL, ESP and Workbench, which apparently no longer exist. All these were of course initially implemented in a DOS environment. Meanwhile, several previously widely-used mainframe packages either ceased to be maintained or only much later were converted, particularly EPS, TROLL and XSIM, the last of which was not. At the margin of what might be classed formally as econometric packages, in the sense of being used by econometricians, various other packages were also developed for the personal computer or converted from mainframe versions. These include such general statistical packages as SAS and SPSS, as mentioned earlier, but also MatLab, some aspects of the use of which is considered by Herbert in this journal issue [157]. Other near relatives include GAMS, which, among its characteristics, provides support for modeling linear, nonlinear and mixed integer optimization problems. But it is difficult to describe the history of the development of such programs in the present context without carrying the story far afield.

Focusing on econometric software, there is some degree of minor controversy as to whether the porting of a number of the previously mainframe programs to the PC were "straight ports" that took little or no advantage of the characteristics and potential of the PC. This debate's full ramifications are best left to after-hours gatherings of the cognoscenti. The significant feature of the microcomputer for econometric software development is the combination of a machine on every desk and a machine on every desk, in much the same sense that house price is said to be determined by location, location, location. The simple ability to acquire a machine that even in its early days offered the capability to perform nearly any task the econometrician might conceive, if not then, certainly in the near future, obviously provided stimulus to econometric software development. However, there was also a "push" factor in many cases. The turnaround time associated with mainframe batch processing has been mentioned. But even in the case of time-sharing systems, mainframes posed problems for their users. At Chase Econometrics in 1983, for example, it was not unusual to watch an economist type in a sequence of commands at a terminal, one by one, then in each case wait for even as much as a half-hour or more for the machine to respond, particularly during busy times of the day. The original PC was not the equivalent of the mainframe in processing power, but neither was it shared. As a consequence, given the improvement in the PC's facilities during its first three years, it was certainly well within that time that this microcomputer became the effective equivalent of a share of a mainframe. An aspect of this distinction is provided by Stephen Fried, who in 1983 related: "a PC user recently discovered that his CRAY [super]computer executed one of his applications a factor of 180 faster than his PC. But because the CRAY was serving 100 users, the [actual] turnaround time was only a factor of 2 better than the PC" [110, p. 197].

The fact of the rapid adoption of the PC, particularly in the widespread way it occurred, starting in about 1985, clearly provided a potential market for econometric software. This market was and is small in comparison with the market for Excel, inasmuch as the number of economists worldwide is certainly less than 100,000, however loosely "economist" might be defined. But this number is obviously greater than the number of mainframe installations serving any collection of economists in 1980. Moreover, whereas economists in many cases usually had only a minority vote in the selection of mainframe software packages, as a user of a PC he or she has dominant influence, particularly to the degree that the software is purchased personally. What may be surprising, actually, is that the number of software packages listed today on the American Economic Association's Resources for Economists website (www.rfe.org) is not greater than it is, especially in comparison with the number of mainframe econometric software packages in Francis' 1981 compendium, mentioned earlier [108].

Broadly considered, the development of econometric software packages for the microcomputer, and in particular for the PC, can be seen as an extension of mainframe-based development in the 1980s. There was a transitional hiatus in the 1980s, but once developers adapted to the environmental characteristics of the PC compared to the mainframe or the minicomputer, the direction of movement, in terms of more and "better", was unambiguously forward. In the 1990s, once packages began to be converted from DOS to Windows, a hurdle that forced developers to focus temporarily but rather intensively on non-econometric aspects, there was an acceleration in their incorporation of additional econometric techniques. Various new

estimators have been incorporated, including those for GARCH and other processes and methodologies introduced into the econometrics literature since 1980. For many packages, particularly noticeable has been the incorporation of facilities supplementary to the computation of parameter estimates. Test statistics have proliferated: in 1986 the test statistics provided by most packages generally did not differ much from what was offered in 1976, or even 1970. In North America specifically, this accelerated accumulation essentially represents another British invasion, with some flanking support from certain Canadians: Hendry's PcGive and the Pesarans' MicroFit have been in the van of the charge, even if the leading edge may have been a series of British and Canadian econometric methodology papers and monographs [44, 53, 54, 118, 148, 156, 209, 237, 256]. These publications both espoused the ideas behind PcGive and MicroFit and provided critical details, which also influenced the latest crop of econometrics textbooks. The exogenous counterthrust by Granger and his allies encountered little resistance, and as a consequence a great deal of cointegration has ensued [14, 90, 125, 126, 339]. Of course, there are still econometricians who remain to be convinced, as a matter of methodology, but the point here is simply one of software features becoming available.

However, in addition to such time series-related development, there was generally a broadening of offerings. For instance, the development of B34S and LIMDEP illustrates this, each expanding the offerings of probit, ordered probit, and logit. In the 1980s, LIMDEP in particular incorporated the computations described in Heckman's seminal 1979 paper, specifically Heckman's estimator [129, 130, 145], as well as the tobit EM algorithm mentioned earlier. In addition, estimators for McFadden's conditional logit model were added, as well as a limited information maximum likelihood estimator for the nested logit model. Beginning in 1990, discrete choice models became such a well-defined subcomponent of LIMDEP that a now-separately distributed package, named NLOGIT, was spawned. One of Greene's stated aims was to automate "a very large number of related limited and discrete dependent variable models as well as a constellation of nonlinear econometric techniques" [134]. In the late 1990s, major areas of program development have been the incorporation of panel data estimation facilities and the estimation of stochastic frontier models: "The main feature of the package is a suite of more than 100 built-in estimators for all forms of the linear regression model, and stochastic frontier, discrete choice and limited dependent variable models, including models for binary, censored, truncated, survival, count, discrete and continuous variables and a variety of sample selection models. Beginning in 1990, a large and growing segment of the package was built for analysis of data on discrete choice (multinomial logit and probit models)" [134].

Other packages showed similar developmental traits. As indicated earlier, the development of EViews has been entirely on the microcomputer and occurred in two phases. The first incarnation, under the name MicroTSP, was developed originally for the Apple II and later ported to the PC. EViews itself, the development of which began in 1990, was released as an original Windows program at the beginning of 1994; Lilien is the original author, but more recently development of the program has become a collaborative effort [202]. During the past almost 10 years, EViews has progressed far from its origins as essentially a regression program. It offers a number of single equation time series techniques and options, supports VAR models, and now offers support for the development and use of structural models generally comparable to that of packages such as BETAHAT or Modeleasy+. The data base management capabilities of EViews have likewise become much more comprehensive, although not yet competitive with programs that have been specifically developed for large scale data base management [282]. EViews is now quite broad in coverage, much more so than more focused packages such as BETAHAT, FP, LimDep, Modeleasy+ or Stata. It appears to aim to support nearly all current econometric "schools," ranging from Box-Jenkins, to GARCH, to a fairly wide range of linear and nonlinear "traditional" single equation and structural equation estimators, to VAR.

This increase in features and usability is not limited to only the newest programs. As one of the "originals" and with a history of being represented on a wide variety of platforms, including mainframes, minicomputers, and workstations, TSP retains some obvious vestiges of its past. More importantly, the program has consistently been found to provide numerically accurate results and to provide its users with the necessary facilities to perform complex operations in a flexible, user controlled way, features that are particularly important when performing nonlinear estimation [227]. One of the program's salient features is that, whenever relevant, its estimation algorithms use analytic derivatives—supplied by the program, not the user—which partially explains why TSP has been found to provide more accurate estimation results

than many other programs [220]. In native mode, TSP is operated using free-format commands and offers all the standard econometric estimation methods, including ordinary least squares, two and three stage least squares, GMM, LIML, FIML, ARIMA, Kalman filter, ARCH and other time series techniques. It also provides a "roll-your-own" maximum likelihood estimator, where the likelihood function can be evaluated either by a single equation or using a user-programmed procedure. Supporting the estimation facilities, TSP provides a variety of diagnostic and testing facilities, including a procedure to test nonlinear hypotheses. In addition the program also incorporates programming capabilities, including support for matrix algebra and user-written procedures that permit users to apply TSP to the development of new econometric techniques. However, the program now also has an alternative operating mode: it has "joined the Ox-metrics family of packages," using GiveWin as an interactive Windows shell. Essentially, what is implied by this phrase is that GiveWin can now function as a user interface for TSP: in response to user directives GiveWin generates the equivalent TSP commands, which then are processed by TSP itself. Operating in this context, TSP can be run either as a batch program or interactively. In addition, TSP for MAC OS version X is expected to be released in the late summer or early autumn of 2003.

As the foregoing indicates, most econometric software packages tend to provide an intentionally broad range of facilities, and some strive to be econometrically all encompassing. One that breaks this mold is BACC, the principal authors of which are John Geweke and William McCausland. BACC was created in the early 1990s with the goal of making tools for Bayesian econometrics "as accessible to practitioners as are those for classical econometrics." In the beginning it was made available for use in a DOS environment as a standalone program. However, one of the notable features of the software is that in 1999 it was re-issued in the form of a Dynamic Link Library (DLL), which has the effect of enabling it to be "called" from other programs as a set of Bayesian techniques, either instead of classical techniques or as a complement to those techniques. Therefore, it can function as an "add on" to a program that otherwise handles the all the basic data management tasks and provides other such user support that is non-specific to the particular Bayesian estimation and directly related operations. This approach results in a different type of "embedding" than that described earlier in the case of BRAP and B34S, but operationally it obviously has some of the same properties.

Stata in contrast is an example of a program from the statistical software world. Originally designed by William Gould, it was initially developed as a DOS program in the mid-1980s and first released in 1985. It is today being developed by "a team of statisticians, econometricians, and computer scientists" [344]. Available for both the Apple and the PC, versions of Stata exist for multiple operating systems, including Apple, Linux, Unix, and Windows. Since the beginning 1990s, Linux and/or Unix have been supported on a variety of machine types, including the HP/9000, DEC RISC, IBM RS/6000, and Sun (Solaris); in 1995, the program was ported to Convex supercomputers under ConvexOS. Stata is designed to provide data base management, graphical analysis support, and a range of statistical techniques for time series, cross section, and panel data. It is cast as a more general "statistical platform with strong support for econometrics—including panel-data, survival, survey data, and time-series—and complete programming facilities for data management, new estimators, graphics, and GUI development" [344]; this is the selective description of capabilities. The developers of Stata encourage user participation in its development through "journals, email list servers, software archives, and special program features" [344].

Another type of recent development is that exemplified by WinSolve, a program developed by Richard Pierse specifically to solve models of various types, among them both structural and Real Business Cycle (RBC) models, using a variety of different algorithms, including both Newton and Gauss-Siedel. This package can be viewed as competitive with CEF, MatLab and other packages that offer both model solution options and facilities to support optimal control techniques. Its lack of capability to actually perform a regression may raise a question in some minds of its present relevance, but if there was any doubt WinSolve places itself firmly in the context of econometric software by offering the capability to read proprietary data files from EViews, MicroFit, MODLER, PcGive, RATS, and TROLL, among others. The package also provides the capability to solve models so as to incorporate model consistent expectations and other rational expectations schemes. In addition, it incorporates a Stacked Newton algorithm that permits the solution of models such as the Ramsey [272] and Koopmans [182] optimal growth models. It also offers such standard structural model forecasting features as support for "ragged edge" solutions, and

both constant and multiplicative adjustments, including the Type 1 and Type 2 "fixes" beloved of Treasury modelers in the UK.

Finally, although there is no free lunch, there are freeware econometric software packages, among them EasyReg, FP, and TSMOD.[35] Interestingly, these packages exhibit diversity in several respects, ranging from their econometric characteristics, to the source languages used to write them, to the operating systems under which they run. The primary author of EasyReg is Herman Bierens and it is a Visual Basic program, thus a Windows package. It is seemingly comparable in its econometric facilities to the standard estimation packages, but ranks among the less developed packages in terms of its data management and other creature comforts. The authors of FP are Ray Fair and William Parke. FP is a Fortran program that runs under DOS and is closely keyed in its facilities to Fair's books and his models. The author of TSMOD is James Davidson and it can be run under either Windows or Linux, and apparently involves some degree of use of Java. The program is described by its author as "a new project, still very much under development," however at present, econometrically, the program appears to be specialized to single equation time series modeling. These packages are described in more detail in the supplementary compendium, found at the end of this journal issue.

It should be evident from this description of a selection of existing programs that, although the initial conversion of mainframe packages to the PC may have represented at first a step backwards in some respects, particularly during the period from 1981 to 1987—reflecting less advanced operating systems and language compilers, and other such aspects of the 1980s PC environment—by the later years of the decade the forward progress had resumed. Particularly from 1984, the PC offered screen graphical facilities that were vastly superior to those generally available on either mainframes or minicomputers. The RAM available to PC users matched or exceeded that generally available to individual users of either mainframes or minicomputers in 1983, and without the execution delays experienced by individual mainframe users. In fact, by the mid 1980s, the problems posed by the PC for developers were much more the result of an attempt to do more computationally than had ever been done before than a consequence of the PC offering a less enabling environment. Rising aspirations of developers, combined with the increased expectations of users, many of whom had a "Madison Avenue" inculcated view of what the computer was supposed to be, colors many developers' retrospective view of the problems of conversion from the mainframe or minicomputer to the PC.

Of course, there were certain weaknesses. For example, under DOS a number of packages were weak in terms of their display of graphs and other such "visuals." However, in retrospect it is now very easy to see that DOS was a difficult environment in several respects. Reflecting this environment, it might even be appropriate to apply this criticism of weakness in visual displays, in varying degrees to all econometric software of that time. But having identified this problem, what is particularly important to understand is its source: the source of the problem is essentially that a DOS program, and its developers, must take responsibility for each of the peripherals used by it and attached to the computer, either directly or via a network connection, including monitors, printers, and plotting devices. Successfully supporting users who might variously own an Epson, a Hewlett-Packard, or a Techtronics printer consequently required the developer to write specific code for each of these, especially to the degree that it became desirable to make each of these devices sing and dance, which of course it was to their particular owners. This requirement to support specific peripherals individually was of course a distraction to the econometrician-developer trying to provide a suitable set of parameter estimation and other econometric facilities.

In contrast, whatever else one might say pro or con about the Windows operating system, a fundamental characteristic is that it takes responsibility for all the hardware devices used by any computer

[35] The use of the term "Finally" reflects not only the order in which these packages are described in this section of the present paper, but also the fact that their authors provided necessary information about them only after the paper was essentially written. Thus their composite (and brief) description reflects much less their common characteristic as freeware than it does the production deadline for this special issue. Soritec is yet another program current information about which was received too late to include here, but which is described in the compendium.

running under it. The programmer of an applications package, an econometric software package in particular, need only make a tight link between it and Windows for the user to be able to use quite successfully any attached peripheral that is supported by Windows. In this process hardware vendors must have earnest talks with Microsoft, but if these folk form a happy family, the user of the econometric software package can change from one monitor to another, from one screen resolution to another, and still find that the program works. To be sure, certain screen resolutions may provide a more satisfactory result than others, but at least something intelligible normally shows up on the screen. The same cannot necessarily be said for DOS and an arbitrarily chosen pixel graphics monitor. There is a lesson in this, when considering multi-platform software implementations: different platforms provide different levels of developer support affecting differentially the use of the software—to the degree that the developer does not do extra work to counteract this.

The entire saga of the conversion of econometric software packages from DOS to Windows is too lengthy a tale, but certain aspects are pertinent. Recall from the earlier discussion that EViews, the successor to MicroTSP, was one of the first econometric software packages to be converted to Windows. All development of MicroTSP ceased in 1990, and a new program was begun using C++ as the programming language for the Windows environment. The Windows version was released at the beginning of 1994 as EViews. A revealing comparison can be made between EViews, on the one hand, and AREMOS and MODLER on the other that illustrates a possibly general characteristic of software development and user reactions. The conversion of these other programs also occurred during the middle 1990s, but took longer and, interestingly enough, both these programs still have users who are steadfast in their determination to stay with the DOS version as long as possible, notwithstanding that various special modifications have to be made to allow these versions to run satisfactorily under Windows 98 and Windows ME, or Windows 2000 and XP, and notwithstanding the DOS and peripherals problem. One of the reasons for the slower pace of the conversion, and certainly the acceptance of the converted versions by users, is apparently that both AREMOS and MODLER are long established and quite reliable as DOS programs, provided of course that the user owns the "right" machine with a standard peripherals configuration. However, users of the DOS versions fully understand their quirks, know the command languages of these programs, and apparently find that the adoption of the Windows version involves a learning curve they would prefer to avoid. In contrast, EViews is a much more capable, comprehensive, and robust program than MicroTSP. From a long-term user's perspective, other than interface, there is possibly less difference in capabilities between the DOS and Windows versions of AREMOS and MODLER than between MicroTSP and EViews. In short, User inertia is an important aspect of the software experience.

The conversion from a command line to graphical interface is specifically an issue that needs to be considered. Actually, the DOS to Windows conversion need not involve this interface conversion: it would be perfectly possible to write a Windows program that within the program's "window," the visual area of that program on the screen, looked essentially the same as its previous DOS command line incarnation, assuming that under DOS this was the program's interface. The reason that this type of conversion is an unusual approach is of course that users (and developers) expect a Windows program to look like a Windows program. And demonstrably this convention applies even more widely than just the Windows environment. From Apples to Linux to Windows, there is a "right" style and that style is now broadly the same across all these operating systems, at least in comparison to a DOS command prompt. Today, most people demand a WIMP interface—Windows, Icons, Menus, and Pointing devices—because there is now a near-reflex reaction against command line interfaces.

One of the commonly perceived defects of a command line is that it requires abstract knowledge: for a particular command, there is no button on the screen to "click on," thus the learning curve may be sufficiently steep as to repel absolutely any new users. Part of the reason for this repulsion today is simply that this interface lacks visual excitement; ostensible modernity is a powerful molder of taste and opinion. However, as an antidote to embracing unreservedly the WIMP interface, consider, for instance, the implications of trying to use a word processing package by successively clicking on scrollable lists of words in order to form sentences and paragraphs, rather than typing them letter-by-letter. Obviously, typing is better; but this conclusion of course reflects the presumption that people already know both the letters and most of the words they might want to use. And this knowledge is of course also the reason why, in the

case of word processing programs, the menus and icons appear at the window margins: they supplement the principal data entry process, the typing of the letters and words into the main, usually white, space of the program.

In the case of econometric software, there is unquestionably a role to be played by menus and pointing devices: certain types of operations, such as selecting various default settings or working with graphical displays, are particularly amenable to "point-and-click." However, specifying equations may be an activity more like word processing. In addition, the case can be made that even for menu or click-appropriate operations, a command line alternative should be offered as well: econometric research often involves repetitive operations performed at regular intervals, such as making certain data transformations, estimating or re-estimating equations, and the like. Such operations may be able to be performed interactively, but after a few repetitions it becomes evident to anyone that simply executing a macro once a week or once a month is preferable. During the initial DOS to Windows conversions, the WIMP interface was sometimes seen as a be all and end all, but as time has gone on, there has been a steady and perceptible move back towards at least partial command line operation. In the context of econometric software, a fundamental reason is that the command line is the basis for allowing a user to specify the precise operation that he or she wishes to perform.

Much more could be said about these particular issues, but these comments have been made principally to illustrate that interface design plays a part in econometric software development that should not be ignored, especially to the extent that the program is created for other people to use. Another inference to draw from this description is that conversion to a new operating system environment is a significant step in the life of a program. Furthermore, the Windows environment has two very important additional characteristics. First, it is not static: Windows 3.x, Windows 9.x, Windows ME, and Windows XP are related operating systems, but *not* the same operating system. Each change in operating system insures that some program feature that worked before will not work now, at least as advertised. Even in those cases that successive operating systems represent improvements, the change nonetheless imposes the need to change the applications programs, if only to take full advantage of the particular new features of the new operating environment. The second important Windows characteristic is that most machines that run Windows also run MS Office, as a matter of course. In part because Microsoft makes Microsoft Office, as well as Windows, and because users change their versions of Office when they change operating systems—and use Office in preference to its competitors in many cases—in practice there is a requirement for a software developer both to make econometric software packages work with Windows and Office and to change the versions of their packages whenever Windows and Office change. It is also not insignificant that government statistical agencies and other providers of the data that econometric software packages use commonly make that data available in Excel spreadsheets. Such contextual issues are important in explaining why it is that it took most of the 90s for econometric software developers to adapt fully to the change from DOS to Windows, assuming that they have fully adapted.

The degree to which the PC has progressively become more competitive to the mainframe is an additional important story. Lest there be any question, there is good evidence from the LINPACK benchmark results [73] that the RS/6000, the IBM workstation machine of the early 1990s, was faster than the Cray SV-1 supercomputer, offering peak performance of 1800 Megaflops per second, compared to 1200 for the Cray. However, the current 2 gigahertz Pentium 4, which is available on a number of notebook computers, offers peak performance of 4 Gigaflops per second [72]. Furthermore, notebooks with 60 GB hard disks are not uncommon and some notebooks sport screens that offer as much as 1600 x 1200 pixel resolution on the notebook screen, not to mention 2048 x 1536 maximum on a suitable monitor that might be attached. Some of these machines in addition support 100 megabit or better network connections and may incorporate both Bluetooth and WiFi (802.11x wireless networking). Obviously, not every user has a machine with these capabilities, but they will have, and in the next year or two, in the case of most users. During the past 20 years, the PC has changed from a machine, in terms of peak performance, that was 100 times slower than the Cray SV-1 to one that is now more than 3 times faster and fits in a briefcase.

3. Econometric Practice: the Econometric Software Package Today

Classically, econometric software packages have taken the form of programs that offer a specific set of selectable options. This was particularly true of regression packages of the 1970s and 1980s, and it is still true of Autobox, CEF, MicroFit, REG-X, and other Windows packages that fundamentally offer a "point and click" user environment. It is less true of packages such as AREMOS, B34S, LIMDEP, MODLER, RATS, and Stata, among others, which each incorporate a programmable command language that permits the creation of macros to several levels of nesting and, to varying degrees, offer features found in computer languages like Basic, Fortran, Java, or Pascal. Such a facility allows the user to perform operations that either constitute composites of the program's standard offerings or extend those offerings. Sometimes, such as in the case of the DOS version of MODLER, these embedded languages have even permitted the generation an interposable interface that changes fundamentally the program's external appearance, allowing it to do such things as display on the screen a tailored menu rather than a command prompt, offer expert features to less knowledgeable users, and otherwise control the overall operation and look of the package. Both the Economist Workstation, a mid-1980s joint product with DRI, and the later PowerStation—offered with text in several human languages by consulting firms in Europe and the US—are examples of such MODLER implementations [15].

However, in order to be able to evaluate properly the implications of such developments, as just described, it might be useful to stop briefly and ask: how exactly did we get here and what does it mean? This approach risks the danger of a pseudo-historical, somewhat Hobbsian or Rousseau-like explanation, complete with early econometric software developers cast as rather primitive creatures, noble or otherwise. Nevertheless, the original stimulus for the development of such macro programming capabilities can be seen as arising from the need to perform various tasks repetitively in an interactive setting. The obvious disadvantage of purely interactive processing, compared, say, to using physical decks of cards, is that commands once processed no longer exist—unless of course these are saved in a file on some storage device and then later executed, or re-executed, by reading that file as a batched set of commands. In the simplest case, a macro is no more, and no less, than a file containing commands of some sort, used in this way. However, there is commonly a need to perform the same transformation on a number of different variables, or flexibly to create moving averages or other such transformations involving lagged values of variables or the like, so that a natural next step is to define a programming syntax that will permit such a macro file to begin to take on the characteristics of an interpreted program, which is then executed by the command processor of the parent econometric software package. Developed in this way, the parent program's macro language progressively becomes a means of symbolically processing variables as data and is enriched to the point that it takes on the characteristics of an actual programming language. The motivation for such developments, by implication, is obviously to improve and enhance the usability of the parent package, so that this type of development has tended to occur in contexts in which the stress has been placed on the broad usability of that program. A good example of an econometric software package enhanced in this way is XSIM: XSIM, having a particularly well-defined macro language, became capable of supporting quickly developed, quite sophisticated macro-based programs to perform specific tasks, utilizing the facilities of the parent program, that simultaneously also simplified the use of the parent program, by providing task-based solutions to particular users' needs—in very much the same sense as today a macro can be used to operate Excel, giving it a temporary persona, or new menu entries can be added to Word for Windows, for example permitting it to generate PDF files or execute another program to fax the document just created, or perform any of a variety of other such user-determined operations.

However, this characterization of the logic of the development of a program's command language is appropriate only in certain cases. In contrast, with the development of Gauss since the mid 1980s and much more recently Ox, another type of package has emerged, essentially an econometric software package that explicitly promotes itself first and foremost as a high level programming and matrix language, and which thereby potentially permits the user to define new econometric constructs. LIMDEP, RATS, Stata, and TSP, among others, also present themselves as sharing this last characteristic, although perhaps not so much as a leading concept. On his Internet "work page," Jurgen Doornik, the principal author of Ox, describes it as "an object-oriented matrix language with a comprehensive mathematical and statistical function library. Matrices can be used directly in expressions, for example to multiply two matrices, or to invert a matrix. Use of the object oriented features is optional, but facilitates code re-use. The syntax of Ox

is similar to the C, C++ and Java languages. This similarity is most clear in syntax items such as loops, functions, arrays and classes." Considered on its own out of context, Ox might not qualify as an econometric software package, being only potentially applicable to econometric problems, but its econometric nature becomes apparent when such Ox "applications" as G@RCH and PcGets are taken into account [154, 194]. TSMOD is another Ox application, as are also ARFIMA, DPD, MSVAR and STAMP, each described in the econometric software compendium at the end of the current journal issue. Similarly, CML and GaussX, also described there, are examples of Gauss applications. In addition to Gauss and Ox, there are of course other programming language packages, including Mathematica, MatLab, R, and S-Plus, that can be used by econometricians for this type of development [9, 21, 353]; MatLab in particular has been used by economists interested in control theory [157]. MathStatica, also described in the compendium, is an "add-on" to Mathematica.

An obvious implication of the econometric programming language approach is that an econometrician using one of these packages no longer necessarily needs to wait for some developer to add a new econometric technique. Within limits, he or she is able to do accomplish this directly. Of course, it has always been possible to learn to program [50], but an important aspect is the "high level" nature of this environment. In effect, a program like Gauss or Ox provides an entry point that does not require the econometrician to learn how to program at the more fundamental level of C, C++, Pascal or Fortran, each of which *computer* programming languages involves the need to program with reference to the operating system. It might even be appropriate to see this high level econometric programming language environment as providing a richer approach to the concept mentioned much earlier, that of permitting the economist to approach the computer screen as he or she might a sheet of paper, but with the operational consequence that what is symbolically represented actually occurs. If you will, AREMOS, MODLER, TROLL and XSIM are examples of econometric *modeling* languages (EML) [286], but Gauss, Ox and possibly other similar packages are effectively econometric *programming* languages (EPL). The critical distinction is the object being worked with: an econometric modeling language has as its objects specific, well-defined types of estimators, time series variables, model equations, and models, but also variable transformations, which in a data base maintenance context are simply formulae that generate other variables and in a model context may be identities. In contrast, an econometric programming language offers as objects matrices, vectors, operators, implicit functions, a looping syntax, and a particular grammar, among other characteristics.

Evidently, econometric software can now be classified into standard estimation packages, often menu oriented, that provide an economist with the ability to perform a given set of operations that are specifically defined by the software developer: the user of such a package *selects* from a set of options. 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 control how particular operations are performed: he or she must put equations together, by combining variables and operators, and by so doing *build* a model. These models can be *solved* or *simulated*. The results can be *plotted* or displayed as tables. Advanced estimation packages (AEP) that both offer a selection of choices and incorporate a macro language capability should also be included in this classification. 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.

In practice econometric programs may not be so neatly classifiable. In order to probe the state of the art, using the existing programs as observations, it is possible to consider a taxonomy that establishes as aspects three central characteristics: Interface, Language Orientation, and Purview. The types of Interfaces can be categorized as:

- Menus and Icons (the now classic graphical WIMP interface)
- Menus and Icons Can Generate Interactive Commands
- Interactive Command mode

- Menus and Icons can invoke Batch (Macro) Commands
- Batch Operation of a Macro

The successive differences are as follows: in the Menus-Icon case, the program is operated directly by the user selecting a menu item or icon, which in particular instances may cause a dialog box to appear, but always involves a direct link between this interface and the operating structure of the program. For example, this method is essentially how Word and Excel operate in interactive mode: click on the open file icon, a list of available files appear, double click on the name of one of these. The file is opened for access, and its contents (usually) are displayed on the screen, at least in part. In the second case, the critical difference is not necessarily what appears to the user, but how control is effected: the menus and/or icons, when selected, generate one or more (usually at the time hidden) commands that actually operate the program. That is, the commands directly cause the operation of the program, but are generated individually, in response to a particular menu or icon selection, and the program responds in "real time" as each command is issued. In contrast, when the program is operated in the third mode, by Interactive Commands specifically, the commands are issued by the user explicitly, without touching either a menu or an icon, and the program responds to the commands individually, as these are issued. The fourth case is distinguished by the generation of a file, a macro file, containing a set of commands, which are generated using menus and/or icons, but this file is then "executed" as a "batch" operation, with the commands then causing the program to perform a sequence of actions, generally without further user intervention. Finally, in the case of the Batch Operation of a Macro, the user writes this macro file, employing Notepad, Wordpad or some other text editor, which in the case of some programs may be built into its interface. Once the macro file has been created, the user submits it for execution. The macro itself directly causes the program to operate, as the program successively reads each of the included commands, generally without further user intervention. As shown in Figure 1, which displays the responses of the developers of the programs listed, it is common for individual econometric software packages to offer two or more of these interface modes. Another, much less common, mode involves the generation of a command file by one program that is then used remotely by another related program, as in the case of MODLER and Stata,[36] or that controls the operation of the other program, as in the case of GiveWin and TSP, or B34S and BRAP, as described earlier.

The Language Orientation of the existing econometric software packages can also be displayed in tabular form, but the issue of Language Orientation involves somewhat greater subtleties. The first language classification characterizes a econometric software package as having the property of allowing its users essentially to select among a given set of specific econometric and related options. Fundamentally, the critical element of language in this context is the objects of the commands: the objects may be an OLS, 2SLS, or other distinct regression command. Alternatively, the object could be a command that opens a model file or causes the solution of a model. Note that here, as discussed earlier, the objects of the language category are types of regressions, equations, models, and the like: essentially econometric entities that are specified by the user in order to cause the program to perform a series of quite well-defined operations. Of course, the specific computations may in fact be unknown by the user, inasmuch as algorithmic details are normally not published by individual developers, although they are generally understood to correspond to known econometric techniques or processes.[37]

---

[36]In the case of MODLER, plot files and other such macros can be used to generate the same plots as in MODLER in the context of both other MODLER "family" programs, such as DataView and MODLER BLUE, and a specific unnamed plotting program that was used in past years to permit plots to be generated by other microcomputers, including Apple machines, used to support dedicated commercial publishing software for the production of very high quality reports [316,337]. In the case of Stata, the process works in reverse using what is called an automatic "do-file" or "ado file." It is this facility that lets users add commands to Stata that appear as built-in commands without having to load a "package" at run time [343].

[37] It is of course common for developers to describe, in standard textbook notation, the operations ostensibly performed. MicroFit, for example, is accompanied by a thick, and very useful, manual that describes in detail the econometrics of the package. Similarly, particularly in recent years, Hendry and Doornik have put considerable effort into the documentation accompanying PcGive, Hendry indicating that "we see our `PcGive' books as textbooks with a strong computational bent (and use them that way), rather

	WIMP Interface	Icons/Menus Can Generate Interactive Commands	Interactive Command Mode	Menu/Icon Can Generate Batch (Macro) Commands	Command Operation of a Macro
Independent Econometric Software Packages					
AREMOS	X	X	X	X	X
AutoBox	X	X		X	
B34S			X	X	
BETAHAT	X		X		X
CEF	X			X	X
EasyReg	X				
EViews	X	X	X	X	X
FP			X		
gretl	X	X	X	X	X
IDIOM			X		X
LimDep		X	X	X	X
MicroFit	X				
Modeleasy+	X	X	X		X
MODLER	X	X	X	X	X
MODLER BLUE	X	X	X	X	X
MOSAIC			X		X
NLOGIT		X	X	X	X
PcGets	X	X		X	X
PcGive	X	X		X	X
PcNaive	X				
RATS			X	X	X
REG-X	X				
SHAZAM		X	X	X	X
SORITEC		X	X	X	X
STAMP	X	X		X	X
STATA	X	X	X	X	X
TROLL		X	X		X
TSP			X	X	X
WinSolve	X			X	X
WYSEA			X	X	X
Econometric Programming Libraries					
BACC			X		X
MLE+			X		
Econometric and Mathematical Modeling Languages					
Gauss			X	X	
LINDO/LINGO		X	X		
Ox		X	X	X	X
Econometric Programming Language Applications					
ARFIMA		X			X
CML					
DPD		X			X
G@RCH		X			X
GaussX	X				X
mathStatica	X	X	X		
MSVAR		X		X	X
TSMod32	X	X			X

Figure 1. Interface Characteristics of Econometric Software Packages

However, the "language" as such does not need to be enunciated: in the particular case of econometric software operated interactively by Menus and Icons, the program's user language per se is actually an abstraction. In this case it exists ephemerally, as a shadow of the progressive use of the menus and/or icons by the user. To be obvious, a language normally must involve explicit commands, and a grammar and syntax, and must cause specific actions. But in fact, as just indicated, these commands do not necessarily need to be typed by the user; they could just as easily be generated by the selection of menu items or icons. The essential issue is the type of objects manipulated. However, for purposes of classification, it is nevertheless pertinent to distinguish, as sub-categories, between an implicit language facility, taking the form of menus and icons, and an explicit command language, for although in each case the same operations could be performed and the same objects manipulated, these modes differ in the way a user of a program is likely to regard the program's manner of operation, not to mention its flexibility to be used in either interactive or batch or quasi-batch modes. It would be possible to go further in this vein, to make even finer distinctions, for as discussed earlier there are programs, including both Excel and particular econometric software packages, that permit the user to generate macro commands as a by-product of using menus or icons.

The second language category also incorporates sub-categories that refer to either an implicit—menus and icons based—or explicit command language, but in this case the objects are vectors, matrices, and other mathematical and/or statistical entities. In this context, the language can be expected to be higher level than Fortran or Pascal or C/C+, but nevertheless a form of programming language. The ostensible effect of language at this level is to give the user greater control over the operations performed. The presumption is that the user will now control to a finer degree the specific operations performed by the computer, although in fact this need not actually be the case: it is possible for the package developer to cause the user's commands to be interpreted intelligently and then converted and executed in a manner determined by the developer.[38] As before, the specific computational operations are unlikely to be known by the user, inasmuch as these algorithmic details have generally not been published by individual developers. At issue is not just the user's ability to specify the characteristics of an arithmetic or algebraic operation, which may be supported, but also the way in which various conditions are evaluated, such as, for instance, the degree of convergence in the context of an iterative nonlinear process [226,227].

However, for classification purposes, it can be presumed that the user of this command language at this second category level does not affect the interface that he or she or some other user may see in order to operate the program. Only in the case of the final language category should the available facilities be interpreted to consist of a command syntax, grammar, and structure that has high level objects, but also permits the generation of onscreen objects that may be organized so as to create an interface for the ultimate user of the program—irrespective of whether or not that user is the person doing the language writing or someone else. This capability may involve a higher level language than that of Visual Basic, or Visual C/C+ and other such development environments, but the effect is nevertheless similar in the sense of providing an environment that results or can result in the creation of an application that is operated by menu, icons, explicit commands or any combination of these jointly or severally. Figure 2 displays the language orientation of existing econometric software packages, constructed on the basis of responses by the developers of the programs listed.

The issue of Purview, or the range of types of operations performed and facilities incorporated, is just as subtle as language. For many economists, the essence of an econometric software package is that it will permit regressions to be performed, usually in such a way that this facility can be seen as supporting the

---

than `manuals'." [151] But what is referred to here is the algorithmic *implementation* of the various econometric techniques—that is, the precise way these are actually coded as instructions to the computer.

[38] This sort of intermediation occurs, for example, in the case of optimizing compilers, which are of course designed to recognize the user's coding inefficiencies and then to produce "corrected" machine language code that produces faster execution, but presumably no other functional differences—in terms of the operations performed and the numeric values generated—but this is only a presumption. McCullough, Vinod, and others have recently argued, in effect, that economists and econometricians may be too generous in their presumptions [228,229].

	Operates by Explicitly Manipulating Econometric Objects		Operates by Explicitly Manipulating Mathematical and/or Statistical Objects		Has Programming Language Features that Modify and Control the user interface
	Using Menus and Icons	Using Command Language	Using Menus and Icons	Using Command Language	
Independent Econometric Software Packages					
AREMOS	X	X		X	
AutoBox	X				
B34S		X		X	
BETAHAT	X	X			
CEF	X				
EasyReg	X	X		X	
EViews	X	X		X	
FP		X		X	
gretl	X	X			
IDIOM				X	
LimDep	X	X		X	
MicroFit	X	X			
Modeleasy+		X		X	X
MODLER	X	X	X	X	X
MODLER BLUE	X	X	X	X	X
MOSAIC		X		X	
NLOGIT	X	X		X	
PcGets	X	X			
PcGive	X	X			
PcNaive	X	X			
RATS	X	X	X	X	X
REG-X	X				
SHAZAM		X		X	
SORITEC				X	
STATA	X	X	X	X	X
STAMP	X	X			
TROLL		X		X	
TSP		X		X	
WinSolve	X	X			
WYSEA		X		X	
Econometric Programming Libraries					
BACC				X	
MLE++				X	
Econometric and Mathematical Programming Languages					
Gauss			X	X	
LINDO/LINGO	X			X	
Ox				X	X
Econometric Programming Language Applications					
ARFIMA		X		X	
CML	X	X			
DPD		X		X	
G@RCH	X			X	

	Operates by Explicitly Manipulating Econometric Objects		Operates by Explicitly Manipulating Mathematical and/or Statistical Objects		Has Programming Language Features that Modify and Control the user interface
	Using Menus and Icons	Using Command Language	Using Menus and Icons	Using Command Language	
GaussX		X		X	
mathStatica		X		X	
MSVAR		X		X	
TSMod32	X				

Figure 2. Language Features of Econometric Software Packages

estimation of the parameters of some equation specification that has economic content. Judging the degree to which an operation is "econometric" is obviously as particular as the various econometric "schools." However, within this classification there are clearly single equation versus system estimators: Ordinary Least Squares (OLS), at one extreme, versus Full Information Maximum Likelihood (FIML) at the other. There is also the time series approach that at one extreme consists of the Box-Jenkins methods and at the other, the dynamic time series approach associated with modern econometrics. However, it is also possible to distinguish between these essentially time-oriented methods and cross-section and panel data methods—and also to make distinctions within each of these classifications. Within the time-oriented approach, it is for example possible to consider methods that are associated with a more detailed specification of the associated error process, such as GARCH. Within cross-section and panel data methods, logit, Poisson, probit, tobit, and other techniques can be distinguished, involving fixed and random effects. There are several ways of examining the classifications of estimators and, historically, at least in the case of linear estimators, the most natural for econometricians is to consider progressively the implications of the failure of each of the particular assumptions that together result in the attainment of Best Linear Unbiased Estimates. Today, in contrast, the diversity has become too great for this particular pedagogic approach to be applied as generally as it was in textbooks, say, in 1980.

Beyond the estimation of parameters, existing econometric software packages include several possible levels of support for data base management. At the most basic level, all programs permit data to be loaded into the program. In the 1960s, data management generally took the form of simply reading the data from cards into a matrix with observations on one axis and variables on the other. Today, almost without exception, all packages not only permit the loading of data, usually in a much more sophisticated way than in years past, but also the creation of new variables, at least constant terms and dummy variables. Most also permit new variables to be created by transformation of the variables loaded. Some permit not only transformations to be performed on those data prior to their use in parameter estimation and other contexts, but also provide rather comprehensive data management, to the point in a few cases of allowing tens of thousands or even millions of variables to be managed as the elements of a data base. Programs, such as EPS, MODLER and XSIM, that were developed in a context of widespread use by people not all of which were necessarily economists, were among the first to offer substantial and sophisticated data management capabilities, including in certain cases the ability to embed transformations in regression, plot, table generation and other commands. Increasingly, especially during the past ten years, such capabilities have become much more widespread among econometric software packages generally. However, to the extent that data base management is well developed in the context of econometric software packages, this well-developedness is generally restricted to the management of time series data, as has been considered in some detail elsewhere [276, 282]. An area of remaining weakness is the management of cross-section and longitudinal data bases, which for optimum use should be supported by software that permits the selection of samples from a larger data set, among other operations; only a few attempts have been made to begin to satisfy this need [57, 58, 282].

The general development of "modeling" capabilities is an aspect of purview. Packages that are specialized to single equation techniques almost as a matter of course have nearly always included support for forecasting and even "regression" packages have offered in a similar way the capability to generate out-of-sample projections; this feature was present in some packages as early as the late 1960s, such as versions of the AUTO/ECON package originally created by Ross Preston and Morris Norman. However, as regards the solution and simulation of multi-equation models, with the exception of the still surviving econometric modeling language packages from the mid-1980s or earlier, such as AREMOS, FP, MODLER, MOSAIC, and TROLL, integrated solution and simulation components, including the necessary supporting data base maintenance facilities, have only been developed for other econometric software packages, to anything near the same degree, just in the past ten years or so.[39] In certain instances, as in the case of CEF or WinSolve, particular packages have been developed that are specialized to the solution of models.

[39] Notably, the solution of econometric models is a topic that for the most part is only occasionally considered in the econometrics literature proper. It is generally treated as either computational or as a topic within the area of mathematical economics [71,76,78,100,104,119-121,165,240,249]. Klein's 1974 textbook is one of the few econometrics textbooks to describe the techniques used to solve such models [173], although Hendry's recent text does mention Gauss-Seidel in passing [149].

A fundamental issue here is model size: as the size of the model solved or simulated increases, in particular, a number of design problems arise.[40] First, there is the problem of the organization and management of the equations that form the model, which some packages have dealt with by permitting the user to name the equations and then, using these names, specify a model by generating a file that is effectively a list of such names, and essentially functions as an indexed list that points to the location of the equations themselves. Other packages define models as explicit sets of equations contained in files, individually usually called a "model file." These files may contain the model's equations either as a group of distinct equations, sometimes in ASCII format, or in a compiled, solution-ready form. The strength or weakness of each approach is in part a question of whether the user wishes to focus attention upon model building, considering a number of alternative model specifications, or upon the repetitive use of a given model as an entity.[41] Associated with the creation and management of a multi-equation model is the question of how to deal with the problem of debugging the model: developing a 300 equation model, or indeed even a much smaller model, has some of the characteristics of software development. Although individual equations may on their own appear to be well-founded, interactive effects between equations can cause a model to crash and burn. Even a model that has been used successfully for a long period of time can suddenly cease to solve, particularly under stressful conditions. As a consequence, the capability to monitor the solution process closely is important, but adequate monitoring facilities have not always been included in solution packages. Similarly, once the model has been solved, if it contains a large number variables, say 300 or more, there is the problem of presentation of the results. The creation of such facilities is far removed from the practice of econometrics, perceived as a matter of estimator choice or specification search, but is critical to the capability to use econometric models effectively.

In the present context, these types of issues are relevant not only in themselves, but also to bring to the fore, using a fairly concrete example, the role of software design as an active determinant of the degree of usability of econometric software. Given the problem of designing software to support the construction and use of a large scale model, it is relatively easy to determine broadly the set of activities that might be involved, which then provides something of a generalized road map to guide the subsequent development. For this particular case, salient historical aspects of that development have been sketched out elsewhere in this journal issue [286] and can be seen as having occurred in a consciously top-down fashion that has then had a general impact. In contrast, other econometric software has more often tended to be created bottom-up, with specific econometric algorithms implemented first, and other facilities then added subsequently, each addition purposeful no doubt, but also often selected in somewhat ad hoc fashion. It is thus more difficult in this more general case to draw immediate inferences from past developments so as to demonstrate easily the way in which particular feature choices uniquely affect software usability. Detailed classifications of types of estimators and tests can of course be made as a statement of econometric content, but given the compendium, further description of this sort may not at the moment be necessary.

What exists today is obviously the result of the past 50 years' development and the range and detail of the offerings clearly express the various aspects of current econometric practice. Future developments can of course only be dimly perceived, but it is possible nevertheless to reflect on the fact that changes in econometric practice do occur and with some degree of frequency. Recall Ed Prescott's interest in testing a new estimator computationally in 1970. It happens that the latest editions of both the Greene and Johnston-DiNardo econometrics textbooks have coincidentally reduced their coverage of variable parameter estimation, compared to earlier editions. The 2003 edition of Greene also excludes such topics as certain

[40] Of course, equation complexity—as this increases—may involve similar issues. At present, the aim is not to exhaustively consider the design of an econometric modeling language, but simply to describe particular issues that should be more generally considered.

[41] It should also be pointed out that some econometric modeling language software operates in a different way than either of these approaches [286]. MOSAIC, for example, generates models as—essentially—modules of the software package itself, converting algebraic representations into Fortran source code, which is then compiled, using a Fortran compiler, to produce the solvable model. Of course, the purpose of the present paper is not to describe all the possible varieties and variation of econometric software.

latent variable models, which were included in one or more previous editions. Inevitably, the pressure to keep the size of a textbook manageable causes some degree of churn in the topics presented. In contrast, for the software developer, leaving out previously supported techniques in subsequent versions of the package is not as easy, inasmuch as code changes can be problematic, but at the very least, user guides and manuals can be revised so as to downplay what are considered out-of-date facilities. Both econometric software developers and textbook authors naturally wish to reflect current practice. An interesting question for dinner time debate is what establishes "current practice."

Greene himself argues that it is not econometric software that drives current practice, although such packages as EViews and LIMDEP may affect what practitioners view as "feasible"—as opposed to just theory. Greene asserts that, when considering each new textbook edition, he examines the journals, and interacts with practitioners directly and through conferences "to see what people are doing." As with the famous Supreme Court definition of obscenity, current practice may be hard to define, but can be recognized when seen. In response to the suggestion that econometric software may determine current practice, Greene responds: "saying that software is central to determining econometric practice may be a bit of a stretch. Surely there is give and take. Keep in mind that to say that software leads practice is to claim that developers of software are the leaders in the proposal of new techniques. I am not sure of that" [135]. It is good to think that among econometric software developers there is at least a modicum of modesty, but is he right? Certainly it is the exception rather than the rule that a software developer per se is the proposer of a given technique, but such a person may have some effect on whether anyone but the proposer ever uses the technique. Is it possible that Keynes presciently had in mind econometric software developers when he referred to "madmen in authority?" He was of course known to have a somewhat jaundiced view of econometrics, as well as of politicians.

An alternative point of view is expressed by Steven Hall, who responded to an earlier version of the present paper by noting "I think you raise a very interesting point about what defines current economic practise and my view is that software plays a much greater role than either we realise or than it really should. There are clearly a number of quite distinctive econometric methodologies around, a very clear statement of this is a paper by Adrian Pagan which was given at one of the world econometric society meetings…He pointed out that three main intellectual methodologies were the LSE dynamic modelling approach, the Sims VAR approach and the Leamer extreme bounds approach. In fact there are very few examples of the Leamer approach being put into practise because there is no readily available software to do it, while the dynamic modelling software and the VAR (RATS) software meant that everyone could [apply the associated methods] easily. When cointegration came on the scene a key feature of taking it up so widely was that you could do the Engle Granger 2 step procedure easily in any package. If we had been given the Johansen reduced rank regression first, I believe that cointegration would have taken years longer to catch on, if at all. Another good example is the contrast between the Quandt disequilibrium models that were around in the 70s and 80s and the Hamilton Markov switching. These two models are actually very closely related but the Quandt models never caught on because essentially Dick Quandt never gave out the software to implement them. Hamilton on the other hand gave out GAUSS code for everyone to do it and so created a whole industry. The basic point is that econometric practise is defined by what most econometricians [are easily able to] do. Another example would be the Hoderick-Prescot filter, which has been heavily critiscised by almost every good econometric theorist who has looked at it, but people still use it because its easy" [142].

In fact, the major determinants of current econometric practice is not a debate that needs immediate conclusive resolution. At the moment, it does not really matter exactly the degree to which econometric software conditions current practice. That it affects it at all is the most important point, and there is little doubt that this is true. It is also significant that the economics profession appears to be depending upon a relatively small number of econometricians to provide this software, many of whom are "part time programmers," to use Hendry's phrase [148, p. 315]. Of course, from some perspectives, the thought of "programmer power" is a little frightening. There is occasionally dark talk of "automated, pre-programmed, context-free, instant analyses requiring only a single keystroke." But such a characterization of the situation also tends to suggest that the consideration of software design is the exclusive preserve of some small group of "programmers" when in fact the proper inference to be drawn is that it matters to the development of econometrics as a discipline that consideration be given by economists collectively as to

how best to develop this resource for the future. It should be evident that, at the least, this is not a topic that should be relegated to the back rooms of the discipline.

4. A Brave New World

Much like Monsieur Jourdan, Moliere's character in *Le Bourgeoise Gentilhomme*, who was surprised to discover that for forty years he had been unconsciously speaking prose, we are possibly at risk of discovering the need to consciously consider the effects of computing on our practices and behavior. Today, if a poll were to be taken, it is likely that the general presumption of economists would be found to be that the computer has made certain lines of inquiry feasible, but otherwise that it has had no significant effects on the discipline beyond permitting faster calculations. Such an inference is easily drawn from Alan Krueger's introduction to the Fall 2001 *Journal of Economic Perspectives* special issue on computing. Referencing the play *Copenhagen*, in which Niels Bohn and Werner Heisenberg are portrayed as having consciously discussed quantum physics in a way that Bohr's non-physist wife could understand, Krueger introduces what is billed as a "symposium on econometric tools," with the idea that "If Broadway actors can attempt to explain the Heisenberg Uncertainty Principle to a lay audience, then surely econometricians and applied economists can explain recent developments in econometric tools to the diverse audience that makes up the readership of [*JPE*]." (p. 3). It is not clear from this introduction which is to be regarded as more simple, the tools or the expected audience, but the pervading spirit of toddlers can safely play is unambiguously present: there are no demons in this toolbox, nor any sharp edges.

A closer consideration of the development of these "tools" reveals both important changes during the past fifty years and various ways in which applied economics has been influenced, both positively and negatively. In his comparatively recent description of the use of the computer by economists, Klein portrays that use in the 1950s as being an extension of the use of electromechanical desk calculators [29, 174]. Both reflecting the characteristics of the computer itself, at that developmental stage, as well as that people naturally initially tend to incorporate the new into their lives in an "old" way, in the case described, the computer appears to have been employed for what was then perceived to be the most difficult and tedious part of regression calculations, the calculation of sample moments [29, p. 511]. Once these were calculated, the remainder of the work was done using more familiar devices, including not only desk calculators but also punched-card tabulating machines. In contrast, modern computers, particularly those with today's relatively high-resolution, graphically oriented screens, are not used simply as number crunching devices. They are now treated as having an important role to play in terms of the presentation of multi-colored charts, graphs and images, as well as data management and—finally—performing calculations, which now of course transparently embed any necessary calculation of sample moments and other such intermediate results. At its most advanced, the computer has progressed from being a calculating machine to the means whereby a user can direct the process of analysis, almost in the role of a conductor, with the calculations usually an unseen (and unsung) element in the action.

The early days of computing were also characterized by a shared computer environment, which has not entirely vanished, inasmuch as the Internet today provides nearly instant worldwide contact between colleagues and even family members. But the new shared environment is different: in addition to being asynchronous—not dependent upon a central device—the modern environment is a matter of communication between intelligent devices, as much as anything else. The machine on the desktop, or laptop, is first and foremost an intelligent device. So much so that, as described, the latest generation of notebook computer is now much more than simply competitive with the number-crunching supercomputer of only 20 years ago. Among other features, such a notebook may contain a 60 GB hard drive or more, thus offering the storage capacity of even several mainframes of yesteryear—supercomputers of course characteristically use a mainframe for data storage and pre-computational organization. With this growth of power and resources on the laptop or desktop, almost all analytic processing by economists has migrated to the microcomputer. The mainframe is, in most interactive uses, simply a distributive Internet node, and it might be difficult in practice for most econometric software users to conceive of an econometric problem that would require the use of a modern supercomputer, although computationally intensive econometric applications, beyond the comfortable capabilities of individual microcomputers, still very much exist [74].

However, it is also important to take into account the implications of the change in the fundamental screen display then and now. The modern graphical interface shields the user from contact with the rudiments of computing, intentionally so. And inasmuch as a graphics interface, in some form, is likely to be with us for at least the foreseeable future, if not forever, an obvious question is how to view the early days of the PC, which are now fading fast from collective memory. One option might simply be to view the earlier implementation of econometric software packages in a DOS environment as having been transitional, and therefore essentially of only historical interest. That is, during the DOS era, the apparatus of computing on the microcomputer developed from its initial, rudimentary state to the point of being essentially equivalent to that of the mainframe. In 1993, to choose a date arbitrarily, microcomputer Fortran, Pascal, and other compilers were fully the equal of their mainframe equivalents in 1983. From the point of view of an end user, the computer became smaller and personal, rather than larger and shared. In short, on this view, the mainframe moved to the desktop. Since then, the dogs have barked and the caravan has moved on.

Yet the DOS era remains important notwithstanding. One reason is that it was a period during which interactive and distributed computing unambiguously became the dominant paradigm. Time-sharing, interactive processing was widespread in the mainframe world, but not overwhelmingly dominant, which did affect the mainframe programmer's environment and perspective. Similarly, networked mainframe computers were also widespread: as briefly described, networked, interactive computing existed as early as the late 1960s (and, on an experimental basis, before that)—and in a number of its characteristics was not so different from operating in a networked environment today. However, the intelligence that the DOS microcomputer placed at the point of user contact meant that, almost inevitably, screens became all points addressable, rather than single line oriented; pixels, not characters, became the atomistic screen units. Line oriented file editors became WYSIWYG word processors. In a DOS world, the graphics environment, such as it was, existed within programs, rather than programs existing within a graphic environment, as in the case of the current Apple or Windows operating systems. But, so long as the user learned the command language, DOS provided the capability to operate programs in a flexible and powerful way. Progressively, irrespective of operating system, whether DOS, Windows, Mac, or Linux, removable storage and other critical facilities were placed at the user's point of contact with the network.

But this is the stuff of romance. For the practical, end-user economist or econometrician who simply views the computer as a tool, the DOS era remains important for the simple reason that the DOS command interface still conditions the operation of most, if not all econometric packages. This is in part because of the desirability of a macro capability, as indicated earlier. For one thing, macros, essentially text files containing commands for a program and executable by that program, have the property of being re-usable. They therefore permit periodic re-execution of particular commands or groups of commands. Or the econometrician may wish to perform a set of operations that inherently involve more than a single command, yet can be viewed as being a single thing, a composite. Alternatively, a set of commands can be used to establish context: the process of initially establishing the computational context—the data source or sources, a date range, and other such settings preliminary to a particular "econometric" operation—can be executed as a composite operation using a macro file invoked by a single command, or even a click on an icon. Just as important, certain types of estimators, particularly system estimators, can involve a sequence of commands. In addition, econometric models, in machine readable form, are sets of equations that may be collected into a text file, if not when in use then at least when being transferred from one environment to another. The written or printed page, expressing thoughts in a mathematical and textual format, is a natural environment to the econometrician, and the closest *ex post* analogue to that may be the command macro, particularly when the need for repetition is taken into account. Of course, such macros must be created manually, but it is easy to imagine, if not today, then sometime in the future, a tablet-oriented notebook computer, which of course exist today if only recently, but on which in the future an econometrician might write a series of mathematical expressions, which are then, effortlessly, operated on by the computer.

This ideal is alluring. But although very much the stuff of science fiction and Star Wars, not to mention TV advertisements by the Microsoft Corporation, it inevitably abstracts from reality: to be operational, a symbol representing a variable must refer to a vector of numbers, located somewhere, and located there because of some prior operation. Indeed, there is a danger in such enthusiasm about recent

computational developments—in speaking out loud in such broad generalities: which is that those who happen to overhear may come to think that the commands that populate the macros, that are represented in the menus and are evoked by the icons, occur simply by wishing. Econometricians who develop software quickly discover at the outset that in fact these housekeeping operations take up the bulk of programming time, both when writing the code and when designing how all this fits into the packages, generally and in detail. The algorithms that define the econometric operations form only a small portion of the total package code. To the econometric theoretician, the majority of code is as plumbing might be to a king, perhaps important to well being, but not very interesting. However, what needs to be more generally recognized is that—by marshalling the data, parsing the input, presenting the output and at some point making some calculations, among other operations—it is mastery of the housekeeping that offers the possibility that in the end what is now drudgery can become a pleasure. *The single greatest limiting factor today on the amount of applied economic research that economists can do collectively is the time they spend preparing to do the interesting part of the research.* To alleviate drain on the economist's energy and enthusiasm, and to minimize the costs involved, is a matter of focusing the attention of both economists and econometric software designers—the demand and supply—not only upon particular estimators, particular constructs (such as the Hodrick-Prescott filter, the Dickey-Fuller tests and the like), and the other issues that dominate the econometrics journals, but also upon the mechanisms of data transmission from the government agencies and other original data sources, as well as upon the design of the software interface.

However, at the moment, the environment does not generally exist in which economists can consider collectively such issues, or more broadly how econometric software packages should be designed so as to better facilitate applied economic research. At least part of the reason is the still prevailing professional sentiment that to talk about such ancillary issues is not what economists do. Therefore, these topics are not addressed in the most widely-read journals, seemingly as a as a matter of course. What is addressed instead is the ongoing development of econometric and economic constructs, but in an atmosphere in which these are divorced from the underlying computational environment; as an immediate case in point, see, for instance, the aforementioned *Journal of Economic Perspectives'* recent special issue on computing. There is nothing wrong per se in indulging in happy talk about "harnessing increased computing power for improved statistical tests" (*JPE*, Fall 2001, p. 129-141), but the level of the discussion throughout the literature needs to be more penetrating, particularly as, in the absence of a full and frank discussion of software design, what is happening instead is not only the piecemeal use of econometric software but also its piecemeal development, usually in response to vague perceptions of demand: thus in various small rooms in the dead of night, a few more tools are progressively being added to already ladened econometric software packages, perhaps at the suggestion of some user, but finally at the whim of the software designer, or an assistant, rather than because of a well considered collective sense of what specifically is needed.

Yet, the matter is not quite that simple. The issue is not only choices between particular econometric features or where the data are obtained. One of the most significant issues involved in the design of current and future econometric packages is the fundamental representational characteristics of computer numbers, both in terms of the conversion of packages from, say, the mainframe environment to the microcomputer, or simply when considering the microcomputer on its own. As is well-known to computer scientists and should be known by applied economists and econometricians, numbers in a computer are inherently approximates to the set of real numbers: they both need conversion from base 10 to base 2 and require representation by a finite number of digits, which in many cases (one might actually say, an infinite number of cases) inherently involves approximation. Aspects of the implications of this conversion requirement have been discussed in the *Journal of Economic Literature* by McCullough and Vinod [228], but notwithstanding that journal's reach among economists, the importance of the need to represent numbers in a restricted way may not yet have been absorbed by economists generally. The point is that, to the degree that the computer is the mechanism by which econometric theory becomes operational, this fact of number representation essentially calls into question the way that theoretical econometrics presents itself. If numbers could be represented infinitely precisely in the computer, then the standard statement of the OLS estimator:

$$\beta = (X'X)^{-1}X'Y$$

would have operational significance. However, in reality, this is *not* how the estimator is calculated today in econometric software packages, or at least not in those that define the state-of-the-art. Of course, it can be argued that what these packages do is to replace the steps implied by this formula (and the other relevant formulae) with operations designed to produce a result that approximates as closely as possible the spirit of the written formula—but that is still not the same thing. At some point, at least in the econometric textbooks, it will become necessary to begin to present the computational process as it is performed by the programs that the students are likely to use. When nonlinear estimators come to be considered, presenting the computational facts of life becomes that much more important. To preserve purity of thought, the Victorian bride may have gone to her wedding night unprepared for the unspeakable horrors of the event, but this is no reason to leave the modern student unprepared for what he or she may face in the night.

Coming to grips with the essential nature of the computer and its representational characteristics is an important, and overdue, first step. However, it is also generally important to adopt the right perspective concerning both the history of econometric computation and what is now possible. It is relevant to note that the advent of the microcomputer occurred at a time of controversy among econometricians. Leamer in 1983 [196] crystallized and focused some of the discontent evident in conference proceedings ten years earlier [37], and in Orcutt's earlier presciently voiced concerns [84, 247, especially p. 197ff]. He expressed as well the dismay implicit in the general absence of answers to critical methodological questions [67, 195]. Sims' 1980 attack on structural models as being unidentified and unwarrantedly dependent on identifying restrictions for their estimation, led ultimately to the development of an alternative form of model, the VAR model [294, 304].[42] The Lucas critique, arguably independently foreseen by Orcutt [247] [84] and Phillips [49], of course was published even earlier, in 1976 [207], although elaborations of it came subsequently and only during the 1980s sponsored considerable debate. Finally, but similarly, the contributions that coalesced as the General-to-Specific Methodology, by Davidson, Hendry, Mizon, Sargan, Srba, Yeo, and others, were published mainly prior to 1980, but it was in the 1980s that the methodology began to achieve widespread acceptance in the UK and abroad. More recently, the methods advocated by Kydland and Prescott [190-192], variously described as computational experimentation or as the Real Business Cycle approach, are put forward by them as econometric in nature in the broadest sense [192, p.70], although Cripps and others may have an equal claim to aspects of the idea [51].

A possibly more charitable view of the state of econometrics in 1980 than that expressed by the proponents of these various critiques and methodologies—and particularly certain of their adherents (see, for example, Charemza and Deadman[44] or Diebold [69])—proceeds directly from an appreciation of the inability that existed in the 1950s throughout even most of the 1980s to realize computationally the insights of the Cowles Commission work, and thus to refine, confront, and elaborate it appropriately. The computational advances from the late 1960s and through the 1970s created a situation in which it had become possible, but not necessarily always easy, to make these computations—taking into account all aspects of the process, from acquiring and managing the data, to estimating parameters, to forming and solving models, and then presenting the results. However, it is only today that the doing has become sufficiently manageable that it is now possible to focus attention much more centrally on the whys and wherefores of the process. This argument should not be interpreted as being in any sense a justification of econometric practices of anyone during the past half century, but instead simply as explanation. The accounts of the various sins of "traditional econometrics," as the strawman is commonly called, often betray an astonishing level of ignorance of the actual historical events, such as the degree of use of the Brookings Model to make "forecasts," or even the actual forecasting accuracy of any models historically: there are dozens of complications associated with any scientific evaluation of forecast accuracy. But it can also be argued that, without very careful qualification, it is bogus to attempt to establish the validity of any model or method on the basis of *ex ante* forecasts that are produced by it and/or its handlers.

---

[42] If the idea is not too impudent, the question can be raised whether economists who use it really believe that the VAR representation is the best achievable, in an abstract intellectual sense, or whether the attraction of the VAR approach more reflects the small amount of data required, relatively speaking, and a manageable computational problem, which the structural model certainly did not—for most economists—twenty years ago? To what degree do subsequent innovations in this approach reflect computational improvements during the past twenty years or so?

It also needs to be recognized, as pointed out by McCarthy [212, p. 386], that the Cowles Commission theorists "really did not have a lot to say about model selection." This phrase, "model selection," is quite revealing about the "traditional approach" that emerged in the 1950s and held sway until approximately 1980, inasmuch as "selection" essentially implies that there is something to select from, which of course may relate to the idea that it was economic theory that in the early days of econometrics was seen as providing the specifications to be tested. The more modern term "specification search" does not have the same connotation. There are many things that can be said in retrospect about the model building process of the 1970s, in particular, none of which are quite so telling as the relative poverty of the data base then as compared to that available today. In another context [283] it has previously been remarked that, considering just macroeconomic processes, the past 40 years has been a period of considerable economic diversity (for the US economy, among others) and it is not obvious that this data base has yet been exploited to the degree that it can be. The data base available to any economist in 1968, for example, was neither as extensive as that available today nor as potentially rich in content, notwithstanding any data quality issues [137]. Today's data base is comparatively richer, both in terms of types of data and in the varieties of economic behavior potentially capable of being revealed. For example, the 1970s and 1980s, in particular, displayed price and interest rate behavior quite different from that seen in the 1950s and early 1960s. It is perfectly valid to decry the quality of the data, certainly as compared to alternative measurement states of the world. However, economists cannot truthfully say that they have exhausted the potential information content. And if longitudinal and other microeconomic data are additionally considered, the scope is even greater. This comparative data treasure trove may not be enough, on its own, to will into existence improved software (or careful economic research), but it should serve to provide some stimulus once the potential of the combination is sufficiently widely recognized.

5. The Evaluation of Software and Its Process

It has been said somewhat waggishly that naïve users of software characteristically expect the software to be error free, but that experienced users are surprised by any correct result. As noted earlier, several of the articles in this journal issue evaluate the performance of econometric software packages, and to date, in the economics literature, such evaluation has only rarely been done, especially in contrast with the statistics literature [26, 107-109, 203, 251, 330, 331, 338]. In a *Journal of Economic Literature* article published in 1999 [228], McCullough and Vinod report that their survey of the set of economics and econometrics journals most likely to publish software reviews identified a total of 120 reviews during the period 1990-1997. Of these, they find that only 3 paid any attention to numeric accuracy and only "two applied more than a single test of numeric accuracy."[43] During the period since, following from earlier articles by McCullough and Renfro, published independently in 1997 [217, 283], a more organized evaluation process has been started that has led to the publication of this particular journal issue, as well as evaluative articles in other journals. It is useful to consider some of the findings so far.

The 1999 McCullough-Vinod article considers generally some of the sources of numeric error. The location of this article in a widely-read, essentially non-specialist journal has appropriately gained it a degree of attention it would not have received in an econometrics journal, but has not yet led to any apparent change in the way economists report the use of software. The same year, McCullough and Renfro published a specific examination of a currently popular topic, GARCH estimation [226] , followed by a related article the next year [227]. The important aspect of the first McCullough-Renfro article is not just the specific findings, but also the discovery that, with two exceptions, developers who were presented with

[43] Because of such findings, the stress here is upon numeric accuracy specifically. However, if accuracy is a fundamental consideration, it is also not the only consideration. In particular, it should not be inferred from the present emphasis that software evaluations that do not primarily probe issues of numerical accuracy are per se any less valuable. At the end of the day, there is as much need to consider evaluatively the features and use of econometric software generally. For illustration, see for instance the book by Harris [143], which in an appendix evaluates the degree to which alternative packages make it convenient to implement a particular technique, or the article by Mackie-Mason [208], which compares the use characteristics of alternative packages for a particular purpose.

the findings had difficulty making the programmatic changes to insure numeric accuracy. Moreover, the case examined was essentially the simplest possible case; more complex GARCH representations are not likely to lead to happier findings. Part of the reason for the discovered problems is that the world does not match the ideal: just as there is no reason to expect that a model's specification is necessarily correct *a priori*, there is no necessity for estimation spaces to be nicely regular and beautifully quadratic. The McCullough-Renfro article points directly to the need to improve real world nonlinear estimation methodology, and it is not obvious that there is a simple solution. Meanwhile, the rage is to implement ever more complex nonlinear estimation schemes, as the basis for reported applied economic research results.

In the current journal issue, McCullough [225] once again steps into the ring and applies to a number of econometric software packages a set of tests originally developed by Leland Wilkinson in the context of more general statistical packages. These tests are designed to evaluate such things as a program's algorithms that implement the computation of correlation matrices, which actually have two aspects. First, it can be demonstrated that the specific way in which a correlation coefficient is computed does matter. But, second, the way in which the Wilkinson's test is implemented essentially requires that the tested program stores data in high precision, generally double precision. The McCullough paper presents important results that need to be carefully considered by econometric software developers, but it also raises some general issues concerning both program design and the limits of evaluative testing. Given the inherent imprecision of most economic data, most famously examined by Morgenstern [238], there is a question whether memory space should be devoted to high precision storage (as opposed to calculation in high precision) simply to insure that a package passes a particular set of tests. The bar can always be set too high, when framing evaluative tests, so as to result in too many packages failing. Obviously, it can also be set too low. Therefore, there is a need for some active discussion among econometric software developers and evaluators, and the editors who guard the publication portals, about the appropriateness of any particular set of tests. It is also open for debate whether tests should be created specifically for econometric software packages, or if it is useful to import them from other, albeit closely related, disciplines.

The Bruno-De Bonis study [38] is avowedly a response to the 1999 McCullough-Vinod paper, but focuses on the specific examination of existing packages that permit the estimation of panel data. The authors note the absence of any readily available benchmark for panel data estimates, which essentially prevents the tester from declaring more than that the same results have or have not been obtained for all packages tested, thus limiting the benefits of the evaluation. At the same time, it is certainly true that when all packages subjected to a particular battery of tests generate the same results, this finding inspires some confidence in the underlying calculations. Equally, when packages differ in the results they generate, this finding points to the need to examine why. The Bruno-De Bonis study is also interesting as an example of a common phenomenon: the interaction between developers and evaluators leading to some disquiet, at least on the part of developers [136]. It is common that when software packages are evaluated, the results are made known to the developers of the packages tested. Characteristically, as mentioned earlier, the developers immediately attempt to correct any computational errors revealed. If, at this stage, the evaluators do not report the results, the effect can be that all programs involved now work properly, in at least the tested context, but evaluators gain nothing but the satisfaction of having advanced the discipline. On the other hand, if the results are publicly reported, at least some users may reach the conclusion that, *ex post*, the programs differ in their accuracy.

Of course, this paradox is only apparent. Given that the packages tested are in use by economists generally, it is important to know which versions of those packages exhibit particular flaws. So long as the testing done is essentially fair and even handed, more published information is better than less. But at the same time, the fact that one version of a program significantly differs from another also argues for the importance of any published results obtained using the package being prominently reported. Moreover, this consideration leads also to the argument that information about program descriptions and documentation deserve to be considered an integral part of the normal reporting process of economists' research findings. Hooke's construction of the microscope is normally termed a "discovery" and is usually given the prominence in the history of Biology that Mendel's or Darwin's discoveries are. Galileo and Newton's work with telescopes and glass prisms are basic to the foundation of the discipline of physics. On

several levels, it is shameful that any discipline that prides itself on being "scientific" in its fundamental ethos should not also publish prominently information that allows applied research results reported in the past to be properly evaluated subsequently (not to mention replicated). In this context, talk of space pressures and the need to save paper are entirely misplaced. Equally, those who develop econometric software have an obligation to do their best to document and describe their work to the best of their ability; it is a two way street.

A particularly interesting aspect of Stokes' paper [320] considering the evaluation of two or more programs together is that it demonstrates, among other things, the fact that it pays to look carefully at results, as well as to approach any particular estimation problem from more than one direction, as the title of his paper implies. The most significant finding of this paper is that the properties of estimators critically depend upon the context in which they are applied, and that routine applications may be as likely to involve inherent computational problems as those of greater ostensible complexity. The paper furthermore demonstrates the importance, after the fact, of a full reporting of computational details, at least to the degree of identifying the software used, including also the platform and version used. Stokes' investigation begins with his discovery of a conflict between results reported by Maddala 15 years previously and his calculations made using SAS. For many years his own software had replicated the Maddala finding with the exception of one coefficient which he though to be a typo in the printed paper. However, when the SAS result was found to differ, a detailed investigation of the problem led to the finding that for this particular class of problems, the convergence tolerance used made a major difference. The ultimate resolution of the problem can in one sense be regarded as a matter of software evaluation: an implication of Stokes' paper is that developers need to pay particular attention to detecting and reporting anomalies. On the other hand, it also demonstrates one of the ways in which econometric software development can assist in the further development of econometric theory and confirms the symbiosis. Moreover, inasmuch as Stokes is himself an econometric software developer, the paper additionally demonstrates that the careful documentation of one's own package can be educational.

The final specifically evaluative paper in the current issue is that by Herbert [157], which considers the use of MATLAB in a somewhat broader setting than econometrics as normally defined. MATLAB is an example of a specialized programming language that, as Herbert points out, fits "between the conventional programming languages (such as C++ and Java) and the higher level applications packages." As discussed earlier, in the past 15 years or so, several programs have appeared, such as Gauss and more recently Ox, that in principle permit the econometrician to exercise greater control over the results obtained, at the cost of taking on more of a programmers role. MATLAB is a member of a class of packages, such as GAMS and others, that generally do not support parameter estimation, particularly the variety of econometric methods, yet permit the construction and use of models. Inasmuch as parameters can be estimated in one package and the estimated equations then transferred to a package such as MATLAB, there is clearly an overlap between this program and econometric modeling packages. The Herbert paper does not evaluate the numeric characteristics of MATLAB—particularly issues of numeric accuracy—so much as consider aspects of the applicability of packages like this to economic modeling, to include the application of control theoretic techniques.

Collectively these papers are both interesting and important. Moreover, it is incumbent upon us to recognize that the evaluation of software is as fundamental to the developmental process as is the original coding. Furthermore, it needs to be generally accepted that this is a task that should be mutually shared between developers and users—and not regarded as being just the province of those who independently, actively set out to test. It has been said before, both here and by others [228], that the precise way in which algorithms are implemented can be as critical to the result as the properties of the algorithms themselves, considered abstractly. Just as important, as an inherent characteristic of the use process, how users apply a package can also determine its numeric accuracy, particularly to the degree that complex, especially nonlinear, operations are performed.

It is important to view both the evaluation and use of software in the proper perspective. The original testing of a package naturally occurs as a part of its development, but this "alpha" testing is limited by the developer's imagination, which is already battered by the design and coding process, making it unlikely that such testing will be performed under a particularly diverse set of conditions. Especially in the case of a

newly written program, the context of its use can affect the results obtained: the order in which particular operations are performed determines the specific numeric values located in memory at each point in time, in effect potentially providing the initial conditions for the next step. If the "right" values are there, the program might generate the correct results, but if not, errors may occur. In addition, users quickly discover their own idiosyncratic style of use: rare is the developer who has not discovered that users soon find ways to do things that he or she never imagined, with an effect impossible to conceived in advance. This trait is especially characteristic of packages that can be operated using commands and macros. The corollary of both these circumstances is that independent users may encounter "bugs" that the developer would not.

During the software's creation, the testing cycle begins early and inevitably involves step-by-step testing, inasmuch as the developer obviously needs to know that each component of a package, tested separately, does what it is supposed to do. As the package takes shape, emerging finally as a recognizable and usable program, the developer will then normally attempt to replicate known results. Unfortunately, these known results are relatively scarce. Zellner and others who have proposed estimators have often also provided estimates, but not all econometricians have been so practical. Furthermore, as the evaluative papers in this issue demonstrate, it is beneficial to have multiple tests of estimators and these can be hard to come by. Berndt's 1991 text [27] is almost unique in what it does, although in recent years Greene [131, 132], Johnston-diNardo [169], and other econometrics textbooks have increasingly provided both worked examples and diskettes or CDRoms containing the data. But it is very rare to find parameter estimates in the literature that include both the parameter estimates themselves and a full complement of associated statistics. As recently discussed by Renfro [283], there remains a substantial burden on developers not only to create packages, often without much feedback, but also to determine what facilities and tests to include, and then finally subject all these to evaluative tests.

However, as indicated, the developer's testing, no matter how intensively done, is of limited utility. An additional reason is that any developer, being either an individual or a group of people, creates a package that suits that developer's own work habits. Someone else needs to confront the software independently, before it is possible for anyone to say that the software is to any degree likely to be reliable. This "beta" testing phase is common to all software, and nowadays has been institutionalized. However, when Microsoft releases a beta version of Windows or Office, literally thousands of developers mount the software on their machines. When a new version of EViews, MicroFit, PcGive or TSP is released, the testing frenzy is considerably less. Yet there is actually no evidence that the numeric reliability of, say, Excel is any greater than that of virtually any econometric software package; quite to the contrary [230, 231]. The difference is probably that with econometric software, beta testing may be less widespread, offset by more care on the part of the developers and perhaps more rigor on the part of the subset of users who provide the evaluative feedback.

Finally, once the software is "released" it is, by implication, validated for general use. Unfortunately, this is the common misperception. In the case of *all* software, not only is a just-released package not yet validated, but over time, with each new release, there is always constant change in the facilities offered, which can result in options that worked previously no longer working, or working badly. Furthermore, just as with houses what matters most is location, location, location, with software it is context, context, context. It is possible to say now, with some confidence, that after 20 years of general use WordStar 3.3, which dates to 1982 or thereabouts, and has not been changed in the meantime, is very nearly bug free—or at least the bugs have become known "features" and can be either adored, for their charm, or avoided. But this is a simple program, in modern terms, and rather limited in its facilities; today, it is also not much used. Microsoft Word 2000, in contrast, remains something of an adventure.

6. Demand and Supply: the Market for Econometric Software

As with any product, the initial cost of developing an econometric software package for widespread use essentially consists of the direct development costs plus the marketing and sales costs; in common with most other software the in-production variable production cost is a relatively smaller consideration. It is obviously possible to continue in this vein and to talk in terms of a market for this product; and among economists what could be more natural than to talk of markets—indeed, there is an almost immediate

tendency for academic economists, in particular, to describe econometric software as "commercial" and as being produced by "commercial vendors," and there are certainly some of those. However, the specific characteristics of this market should be noted at the outset. A fundamental one is that, with only a few exceptions, the programs are each created and maintained by at most a handful of people, and in some cases one or two, not as "large-scale" commercial propositions—particularly in contrast to Adobe, Corel, Lotus or Microsoft, the econometric software "industry" is a cottage industry. Furthermore, the extent to which this software should be viewed as being created, maintained, and sold (or leased) in a free market environment is not obvious; for instance, a number of packages benefit from a variety of both explicit and implicit subsidies. Considering the demand side, an obvious characteristic is that the market is small. As suggested earlier, the number of economists worldwide is unlikely to be in excess of 100,000, howsoever "economist" is defined. Of course, the limit of this market might be viewed as being somewhat larger if undergraduate students, taking an economics course, are included. On the other hand, it is certainly smaller if it is regarded as being limited to "applied economists," or just econometricians. But whatever number is attached as an approximation, the market is obviously not a mass market. Finally, if a value judgment might be permitted, it can be argued that for the good of economics as a discipline, this market should ideally be vibrant and innovative, whatever its size—and, in addition, if might be further argued that those on the demand side have an incentive to prevent market failure.

In the Windows environment especially, there are significant constraints on cottage software development for widespread use. Hendry [208, p. 315] has described the difficulties as including the development of code in a menu-driven, graphics-oriented interactive world in which users have both grown "accustomed to excellent and powerful general packages for word processing, databases, and spreadsheets, and [demand] similar quality products in econometrics. A part-time programmer [cannot] easily attain the required standards," particularly as users demand "superior and detailed documentation, including explanations of the econometrics as well as the software." Against this backdrop, it is not out-of-place to suggest, as a related consequence, that the discipline of the market threatens econometricians with the possibility that at some time in the future the sole remaining regression-capable program could be Excel, which has yet to achieve either a range of parameter estimation methods, a satisfactory set of supplementary statistical tests or even an adequate level of numeric reliability [230, 231]. Excel does in fact permit regressions to be performed, so that arguably it is a market participant, and there are countless examples during the past 10 years of the operation of the dictum that the "good is the enemy of the best." Possibly, and fortunately, the limitedness and specialty of the econometric software market may actually be a preserving characteristic against the onslaught of this giant, as it was with SAS.

Considering the econometric software packages themselves, it might be possible to view them as each having specific traits: the compendium of existing packages included in this journal issue provides a means of directly comparing and contrasting them. It is evident from this compendium that only to a degree are they close substitutes for each other. In particular, the market is not a "toothpaste" market: Colgate versus Crest, both offering essentially the same thing that competing marketing departments attempt to represent as something quite different. For one thing, the philosophy that specific packages embody creates actual differences: for example, PcGive can be viewed as reflecting a different approach to the representation of economic phenomena than does Stata, in fact a different philosophy. MicroFit is more specialist in orientation than EViews. And insofar as econometric software packages are used in an academic context, the fact that packages are philosophically competitive may be a reason to license both, rather than one in preference to another.

Apart from philosophy, there are several reasons why packages differ. One of these is the disincentive on the part of any one developer to offer every facility that a user might demand. In some cases, this disincentive might originate in the developer's perception that there is an infinitesimal benefit-cost to offer, for example, every regression option ever described in the literature; this restriction to particular techniques may therefore cause specialization. Or it may be that certain facilities simply are not logically compatible with others, to the point that inclusion of both adversely affects the program's overall design. The initial historical choice of particular techniques thus may maintain specialization subsequently: for instance, packages that support the construction and use of large-scale macroeconometric models, and are viewed by their developers as specialized to this, may be potentially less likely to contain many options very closely associated with single equation models or, perhaps even more, with cross-section micro data.

For these reasons, the presence of distinguishable and persistent market segments might be viewed as an inherent characteristic of the econometric software market.

However, there is also a countervailing tendency: from the user's perspective, diverse facilities within a given package may make a great deal of sense, at least *prime facie*. Just as with newspapers, where the cost of the paper is not really the purchase price, but rather that of the time taken to read it, the cost of maintaining a library of econometric software packages today is not so much the license fees to obtain them, but the time it takes to learn to use them. Among mass market products, the aforementioned Excel might be a good example of a package the users of which try to apply it universally, rather than employ other programs for specific purposes. One of the consequences appears to have been the demise of a number of specialist programs: for instance, who now remembers Harvard Graphics? Similar behavior can be observed with respect to econometric software packages: a screw can be driven by a hammer as well as a screwdriver, although not always with as good effect. Recognizing this market imperative, a developer might therefore be inclined to try to preserve his barriers to entry by offering a somewhat larger set of facilities than otherwise, so long as the additional ones are viewed by developers and users as representing current econometric practice. The market segments may be fuzzy sets, or it might be that certain techniques are technically similar, notwithstanding an ostensibly different application, thus permitting broader coverage than would be the case otherwise [18, 147].

An important determining factor is of course developers' perceptions of market demand. In advance of the marketwide penetration of a particular new technique, to a software developer there is usually not much feedback favoring its adoption. The first indication of the technique's existence can be an overture from its proud proposer that it would fit very nicely into the developer's package. Sometimes this suggestion will be accompanied by off-prints, algorithmic suggestions and occasionally machine-readable test data. However, once a particular technique is established in the cannon, users may demand its inclusion and developers may include it even in the absence of evident user demand, simply to avoid being caught short. Alternatively, the diffusion of techniques can occur endogenously when a particular developer acts as a promoter of those techniques, as has occurred most famously with PcGive. Exogenously, as the earlier quote from Greene implies, techniques can be added by a developer who, sniffing the air, determines the next new thing. In fact, the birth and death of implemented techniques might make an interesting study, but at present would be difficult to pursue if for no other reason than that econometric software packages are not well documented in the literature.

Of course, comparative software reviews have been published from time to time during the past twenty and more years, but reliance on these is likely to be misleading: the techniques actually implemented in each program, as opposed to what the developer might have indicated, have only seldom been independently verified by compilers of such reviews; vaporware is not unknown in the econometric software market. However, once published, comparative reviews sometimes become a guide to the developer interested in keeping pace, notwithstanding the disinformation they may provide, and thus *ex post* may be to some degree self-validating.

A careful consideration of the characteristics of the existing packages reveals certain distinct categories and these classifications may provide some insight into the design principles of at least some packages. Autobox will never be confused with MODLER, nor LimDep with MicroFit, nor PcGive with RATS. However, relatively few developers have gone on record to declare their particular design intentions and it is therefore not clear to what degree the design-undeclared econometric software packages are simply ad hoc accumulations of techniques. The relative restrictedness in the number of possible users has definitely had an effect on the facilities offered, particularly in the case of the more commercially minded developers. All other things equal, small markets of course do not support a high degree of specialization. Developers who wish to hire assistants may need to boost revenues and consequently need to provide a greater number of techniques in order to attract more users. Obviously, to the degree that there is doubt among users and developers as to what constitutes econometric practice, some confusion ensues and consequently there is a greater tendency for developers to add marginal techniques just in case. There is presently evidence that this has occurred, to at least some degree. Of course, it also must be borne in mind that the market for econometric software is not isolated: for example, particular econometric techniques have been developed for and by those interested in financial theory applications, and with the

relatively recent comparative growth of this specialty, some econometric software packages have stretched to cover on the one hand structural models, on the other VAR models, and on the third GARCH and other specific financial econometric techniques [235, 236].

To be sure, as indicated earlier, users are sometimes inclined to view more as being better. And this attitude extends to other aspects of the software, including for instance the "platforms" on which a given package is available. It is not uncommon to find a particular program presented as being available on Apples, PCs, minicomputers, and mainframes, at least in years past. And in one sense this availability undoubtedly exists: for the right price, developers generally stand ready to port their programs. Hubris can also play a part. But there are also important disincentives: the truth of the matter is that while multiple platforms in principle provide the user with a high degree of perceived flexibility, for the developer the incremental cost can be substantial. In the context of any knowledge or skill-based activity, there is both a cost and a reduced proficiency associated with generality. For the developer to offer a package on two or more platforms may mean, for instance, either limiting the package's capabilities to only those that can be supported on every platform, or producing a different package for different platforms. Today, a computer that supports Windows can be counted on to offer Internet Explorer as a possible browser. Other machines may not. Similarly, Linux offers some attractive characteristics, but to the developer also represents a step back into a DOS-type world inasmuch as in this context peripherals may once again need to be considered explicitly. Few potential users who advocate movement to Linux then say in the next breath, "I'll pay twice the price you charge for the Windows version, and I'll have 500 copies."

Similarly, the attraction to the developer of being able to offer a more comprehensive set of techniques on any given platform comes at the cost of having to pay for the development of the additional techniques. This cost is proportionately greater to the degree that those techniques lie outside the core competency of the developer. And, in this case, the happy solution is not always to hire another assistant: it is well known that multiplying the number of programmers by n does not decease the time of development by the same factor: restructuring a program to include additional techniques is difficult enough for the person who created the program. Including several additional people in the project involves the need both to take time to train these people and to pay the organizational costs of the increased degree of joint development. In addition, to find another assistant involves search costs, not to mention the problem of trying to fit a square peg into a round hole when the ideal person cannot be found easily: that is, there may be a difference in result between hiring an economist able to program and hiring a computer scientist or other person in the hope of training him or her to try to think like an economist. Particularly in the case of applications that involve concepts that are familiar to economists but perhaps not others, this is an important consideration. First class mathematicians, for example, can find even the notion of a time series emotionally difficult. Computer scientists knowledgeable about relational data bases find it easy to consider multiple-key searches, but sometimes difficult to come to grips with the idea that economic data base systems do not necessarily require the same degree of record-locking as do airline reservations systems. Such expansion problems as these, combined with users' notorious lack of concern about numeric accuracy, at least until recently (possibly), provide some temptation for econometric software developers to paper over the cracks and hope that no one steps there: you may think it possible to maintain an extremely comprehensive package, that is highly numerically accurate, and bug-free, on four different platforms for pennies a year, but if you do, you are probably not an econometric software developer.

The realities of the situation are that most econometric software developers need to watch their costs rather carefully. One of the inferences to be drawn by developers from this consideration might be that a greater attempt should be made collectively to harmonize packages in particular non-competitive areas, especially in those instances that the basic usability of the package is at issue, and proficiency is expected as a matter of course. A particular problem that exists for both developers and users is the availability of data. Considered generally, particularly in comparison with yesteryear, data are abundant and easy to acquire. However, it is ordinarily still quite difficult for anyone to download data from the Internet in a way that the observations become immediately usable. There are of course technical difficulties confronting harmonization, when considering different types of data, say time series versus cross-section, but certainly within categories of data, there is no reason why software developers should not work together to improve the situation.

In the case of data downloads specifically, it remains true today that the process of extracting data from the Internet is still much more longwinded than it need be, especially to the degree that the data are obtained from different sources. As an example, go to the Federal Reserve Board website and try to download the Industrial production index data on the US economy for the period from 1945 to the present day. The data are available as Excel spreadsheets, but it can take all but the most data proficient as much as a week to create a data base containing all, or even a sizeable selection of the available series in a form that would be immediately usable in an econometric software package. The situation becomes worse if an attempt is made to create a composite data base containing data from the Federal Reserve Board, the Bureau of Economic Analysis and the Bureau of Labor Statistics. Currently, there is a tremendous multiplicity of effort among users of such data, which cries out for the adoption of a common approach. It should be possible for a user of one econometric software package to trade data with the user of another package, or for the same user to use both packages, with a minimum of conversion effort. What makes the present situation particularly frustrating is that, given the Internet, it is now not only technologically possible for data sources to make observations available in such a way so as to absolutely minimize the time between the download and first analytic use, but once such a (at least quasi Pareto) optimum were attained, the cost to the original data sources would be essentially the same as maintaining the present system [282, 284]. Furthermore, with the possible exception of Canada, the situation expressed by the example of US data may represent the best case.

8. The Next New Thing

Classically, during most of the past 30 years, econometric software developers have stood ready to provide all comers with a sensible selection of econometric techniques, and for the average applied economist, this continues to be a defensible stance. However, there is an alternative view, alluded to earlier and not necessarily directly contradictory, that every serious econometrician should have at hand a package, such as Gauss, MatLab or Ox (some would include in this list, instead or as well, Mathematica, S-Plus or R), that provides the facilities to program whatever facilities are needed at the moment. It is worth examining this alternative view, and an argument might be made in the following terms: there is, to be sure, the need for facilities that permit the development of new estimators and new statistical tests, just as there is a need for facilities that permit the study of the solution characteristics of various types of multi-equation models in various contexts. But considering the broad range of applications, there is a role for every type of package: different people may wish to drive different automobiles, but to teach a son or daughter to drive, it is not necessary to build a car piece by piece. To win the Grand Prix, it is. As argued earlier, the skill set to program an econometric technique in a numerically accurate and efficient way, and to generate the various associated test statistics, may in the end rule out widespread use of econometric programming languages. More generally, there is a wide range of applications of econometric software, and given the relatively small number of active developers, it makes sense for both developers and users to recognize that there are a variety of contexts in which the software is applied, and a variety of needs for particular types of facilities.

The counter argument is that packages such as Ox, as econometric programming languages, can provide an environment for the development of a series of applications, programmed in the Ox language, that collectively can provide economists with the research tools they need. This thesis has been advanced more or less as a self-evident truth, and several “Ox packages” have been developed by econometricians. One such Ox package is PcGets, which presents itself as “implementing automatic general-to-specific (Gets) modeling for linear regression models based on the theory of reduction, as in Hendry [149, chapter 9] ” [154]. Another is the Arfima package 1.0 for Ox [75]. A third is G@RCH 2.0 [194]. In effect, Ox is seemingly being used in these cases in an analogous fashion to Visual Basic, Visual C+, and other Windows developmental environments, to provide a basis for the creation a series of end user packages that provide specific applications.[44] Conceptually this is not so different from the PowerStation created on the

[44] PcGive, PcGets and the next version of STAMP are all written in Ox, but function as standalone programs, requiring GiveWin as an interface, but not requiring immediate access to Ox itself to run; they are in effect “compiled” Ox programs. In contrast, packages such as Afima, G@RCH, and MSVAR require access to Ox to be used, although G@RCH is shortly to be released as a standalone program.

basis of the DOS version of MODLER, so that it may be difficult for someone who has indulged to be too critical of the idea. But one is reminded of Samuelson's invocation of La Rochefoucauld in the context of another introduction [292, p. 7]: "The experienced give us good advice when they can no longer set us a bad example."

Pursuing this course, Ox sets itself up as a third layer of code: Windows first, the developmental language used to write Ox second, and then Ox. Windows, in this context, might itself be regarded as two layers, inasmuch as the Dynamic Link Libraries (DLLs) that form the Windows API, generally written in C and Assembler, sit on top of the Bios kernel. Each layer may provide an easier environment in which to program, but each simultaneously and successively places restrictions on the freedom of the programmer(s) at each level to control exactly how the coding at that level is done. Furthermore, each layer can introduce bugs. Consequently, the approach is not without its pitfalls. Commendably, in the case of G@RCH 2.0 in particular, the authors have provided some evidence of numeric accuracy, comparing results with the benchmark established by Fiorentini, Calzolari, and Pannattoni [103] and the particular model presented in the evaluative studies by McCullough and Renfro [226] and Brooks, Burke, and Persand [35]. The authors note that "Contrary to EViews, Matlab, and SAS, G@RCH 2.0 hits the benchmarks for all steps to the third decimal (note that GAUSS, MicroFit, and RATS also do)" [194].[45] Inasmuch as these authors apparently are not associated with the development of Ox, these results tentatively validate the concept of third party use, although the discussion in McCullough and Renfro is quite relevant to this judgment and should not be ignored [226].

Similarly, it is possible to identify stages of processing and to argue in favor of an approach that provides users with flexibility and variety. For example, in the case of macroeconometric models, there are a number of different computational tasks that must be performed: parameters must be estimated, models need to be organized, validated, and then solved and validated again. These tasks can be performed by one program, and at least for standard cases, a good argument can be made for the use of a single program for many users. However, as mentioned earlier, from the user's perspective there is a logic that might argue for the ability to use a multiplicity of specialized programs to estimate the parameters of different equations of a particular model, on the one hand, and to solve the model in a multiplicity of different ways on the other. Unfortunately, any attempt made to put this into practice today would run up against the absence of common representational standards. This lack of standards not only restricts users, but it also impedes the development of economics and econometrics as disciplines, inasmuch as the result is to create significant disincentives to potentially interesting research initiatives.

Comments such as these could be interpreted as criticism directed at econometric software developers. However, they should not be. Although recent evaluative studies have identified some numeric problems that are as yet unresolved, those who have taken the trouble to evaluate econometric packages have discovered that developers generally react almost immediately to reported problems. As McCullough and Vinod indicate [228, footnote p. 635] "some developers are remarkably quick to respond to reports of errors and many of the errors we recount were fixed even before our article went to press." It is true that sometimes there are systemic problems, and for this reason it would be better for there to be more widespread testing of the software than there is: software development is inherently difficult. Econometric software programs can contain a million or more lines of source code, and even the simplest can contain thousands. Any single line can contain one or more errors. Errors are therefore to be expected, particularly to the degree that users wish to push the envelope and developers attempt to keep up

[45] In the case of EViews, the implied fault has since been determined to be correctable. The source of the problem is apparently what can be construed as an unconventional way of controlling the setting of starting values in EViews 4. Once this peculiarity is taken into account, EViews apparently produces the correct results. There has actually been more than one instance in which EViews has been reviewed and subsequently it has been discovered that the results produced by the program have been affected by nonstandard user protocols. To provide another example: contrary to standard mathematical convention, all versions of EViews to date apparently evaluate $-x^2$, where x=1, as 1, rather than –1. It can be argued that such "gotchas" should be regarded as being grounds for identifying a program as producing incorrect results, inasmuch as users should be able to expect standard mathematical conventions to be adhered to by the developers of a widely used program.

with the development of econometric theory by enhancing and changing their programs. But the evidence so far is that the most difficult problems lie mainly at the margin.

However, once more it needs to be said that both the likelihood and the incidence of errors implies the need for users of software to identify precisely the software used when publishing books and journal articles. The general lack of computational knowledge on the part of most economists, the relatively restricted opportunities for economists to publish articles dealing with computational issues in tenure track journals, and the seeming unwillingness on the part of editors of economics and econometrics journals generally to require authors to disclose information about computational details has sometimes led economists to present their work as computationally well founded, based upon powerful techniques and facilities, whereas the reality is more akin to smoke and mirrors. There is growing evidence that a careful replication of applied results published in many journals might reveal a disconcerting number of problems [229]. Moreover, this evidence points to the likelihood that the greatest problems can be expected to be associated with the use of leading edge computational techniques, which if true is likely to mean a high correlation with the research reports and articles most likely to be considered as a basis for tenure and promotion.

9. Conclusion and Prospective

The period of development of econometric software spans the professional careers of almost all working economists. At each step along the road, what constituted "the present" at that time seemed to offer both promise and the sense of being at the forefront. With such thoughts in mind, it is interesting to read Daniel Suit's 1963 monograph [324, p. 7] describing the construction and use of econometric models. Writing in 1962, he asserts 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, with the aid of high speed electronic computers, we can use models of indefinite size, limited only by the available data." This is of course a reference to a marvel of that day, the IBM 1620, which now seems decidedly less impressive a machine. Yet the quote also masks in its enthusiasm a deeper reality, for shortly thereafter, Michael Evans was using a Marchant mechanical calculator to solve interactively the mid-1960s versions of the Wharton US Quarterly Model. Some years later, for the Fromm-Taubman policy simulations of the Brookings Model, the Brookings model was linearized by approximation and then solved by Taubman using a desktop calculator. Modern nonlinear model solution techniques, which is to say in this case the Gauss-Seidel method, were not used to solve the Wharton Model until 1968-69. The Brookings Model was not solved in this way until the early 1970s [114-116, 300]. In the light of these computational circumstances, it is interesting to read many years later [44], an assertion that the predictive performance of the Brookings Model was poor, notwithstanding that no version of that model was ever used to make an actual *ex ante* prediction. Or one might turn to Diebold [69] for edification concerning the past, present and future of macroeconomic forecasting.

However, although it is easy now to demonstrate both the limitedness of the technology of yesteryear and, even then, the degree of its non-application, it is nonetheless true that in the 1960s heady optimism was the order of the day. For their part, today's software developers need to keep firmly in mind those halcyon, early days of economic forecasting, when good feelings concerning the apparent success of the Kennedy tax cuts led to a seemingly general optimism that the economy was both finely controllable and its future performance eminently predictable. Of course, these were not the sentiments of those proposing the policies nor of those making the forecasts. A careful reading of the publications of the model builders and forecasters of that period make it clear that at the time they felt that their work was tentative and experimental, that each finished model was no more than an hypothesis to test. Nevertheless this work has been in more modern times excoriated as being Icarian in the extreme.

Today, we believe that we are beyond that, that we are on a firmer footing, and likewise in our hubris we might be inclined to echo Suits that "with the aid of high speed electronic computers, we can use models of indefinite size, limited only by the available data" in order to tout our modern software. It is of

course on the basis of the econometric software packages individually described in the compendium found later in this journal issue, which have been developed during the past ten years—either newly or as an extension of previous work, together with the software being developed today, that the economics profession will in the near term undertake applied research. And in parallel with the economic forecasting experience of the late sixties and early 1970s, there is a danger that flaws, real or imagined, due to negligence or as a result of pure accident, or even caused by other parties, could lead to a sense of betrayal and outrage on the part of users. This is the Democlean sword that hangs above the software developer's head.

The threat is actually broader than simply whether or not the software is accurate and suitable to the task, in the hands of a knowledgeable and experienced user. There is an implied responsibility to produce software that can be used by any economist. As a consequence, software developers, particularly those who set out to develop software for widespread use, need to consider their role. As Johnston has recently noted [168],

> the number of suggested estimation, testing, and diagnostic procedures has proliferated, perhaps to the point where even econometric theorists are not fully cognizant of the nature, advantages and disadvantages of the various procedures, and certainly beyond the point where the average applied econometrician can hope to make sensible judgments about what research procedure to implement. It is thus all too possible for someone to activate an econometric software package, of which he has only a dim understanding, to apply it to data of whose nature and provenance he is ignorant, and then to draw conclusions about an economic situation, whose historical and institutional realities he has, perhaps, not studied in any depth.

And this view seemingly assumes no software flaws. The issues that Johnston raises are too profound to be considered in any detail at this stage, but the implications are obvious. Software developers face the problem not just of developing numerically sound packages containing a well defined, given set of features, but of designing and developing packages that are numerically sound and conceptually well founded in a context in which there is little or no external guidance beyond the lure of the next new thing. Johnston's words are not simply an assessment, they are also a challenge.

References

1. *Annual Report of the H.G.B. Alexander Research Foundation*. 1995, Chicago: Graduate School of Business, University of Chicago.
2. Adams, F.G., *Email re software used in the Economics Research Unit, Department of Economics, University of Pennsylvania*, C.G. Renfro, recipient. 7 August 2003.
3. Adams, F.G. and R. Summers, *The Wharton Indexes of Capacity Utilization: A Ten Year Perspective*, in *Proceedings of the American Statistical Association: Business and Economics Statistics Section*. 1973.
4. Adams, M.O., *The Kentucky Economic Information System: a resource for government, academic, and private sector researchers and libraries*, in *National Online Meeting Proceedings - 1981*, M.E. Williams and T.H. Hogan, Editors. 1981, Learned Information: Medford, NJ.
5. Adams, M.O., *The Kentucky Economic Information System*, in *Exemplary Systems in Government Awards '85-'86: the State of the Art*. 1986, Urban and Regional Information Systems Association. p. 144-158.
6. Aitchison, J. and J.A.C. Brown, *The LogNormal Distribution*. 1957, Cambridge: Cambridge University Press.
7. Allen, A.T., ed. *SAS/ETS User's Guide: Econometrics and Time Series Library. 1982 Edition.* 1982 ed. 1982, SAS Institute, Inc.: Cary, North Carolina.
8. Almon, C., *The Craft of Economic Modeling*. Fourth ed. 1999, College Park, MD: Department of Economics, University of Maryland.
9. Amman, H.M. and D. Kendrick, *Programming Languages in Economics.* Computational Economics, 1999. **14**(1-2): 151-181.
10. Ando, A. and R. Rasche, *Equations in the MIT-Penn-SSRC Econometric Model of the United States*. 1971, Department of Economics, University of Pennsylvania: Philadelphia, PA.
11. Armstrong, A.G. and L.J. Slater, *A computerised data bank*, in *The Econometric Study of the United Kingdom*, K. Hilton and D.F. Heathfield, Editors. 1970, Macmillan: London.
12. Ball, R.J., ed. *The International Linkage of National Economic Models*. 1973, North Holland/American Elsevier: New York.
13. Ball, R.J. and S. Holly, *Macroeconometric model-building in the United Kingdom*, in *A History of Macroeconometric Model-Building*, R.G. Bodkin, L.R. Klein, and K. Marwah, Editors. 1991, Edward Elgar: Aldershot. p. 195-230.
14. Banerjee, A., et al., *Co-integration, Error Correction, and the Econometric Analysis of Non-stationary Data*. 1993, New York: Oxford University Press.
15. Barber, F., *Statistical Software for the PC: PowerStation.* PC Magazine, 1989. **8**(5): 214-216.
16. Barker, T.S., *Economic policy formulation in a large-scale model*, in *Optimal Control for Econometric Models*, S. Holly, B. Rustem, and M.B. Zarrop, Editors. 1979, St Martin's Press: New York. p. 286-299.
17. Barker, T.S., W. Dada, and W. Peterson, *Software developments in the Cambridge Growth Project 1960-1976: the origins of software for space-time economics.* Journal of Economic and Social Measurement, 2003. **this issue**.
18. Belsley, D.A., *Estimation of systems of simultaneous equations, and computational specification of GREMLIN.* Annals of Economic and Social Measurement, 1974. **3**: 551-614.
19. Belsley, D.A., *Centering, the Constant, First-Differencing, and Assessing Conditioning*, in *Model Reliability*. 1986, MIT Press: Cambridge, MA. p. 117-152.
20. Belsley, D.A., *Conditioning Diagnostics*. 1991, New York: Wiley.
21. Belsley, D.A., *Mathematica as an environment for doing economics and econometrics.* Computational Economics, 1999. **14**(1-2): 69-87.
22. Belsley, D.A., *Email re the relation of GREMLIN to TROLL and other issues*, C.G. Renfro, recipient. 22 March 2003.
23. Belsley, D.A. and E. Kuh, eds. *Model Reliability*. 1986, MIT Press: Cambridge, MA.
24. Belsley, D.A., E. Kuh, and R.E. Welsch, *Regression Diagnostics*. 1980, New York: Wiley-Interscience.

25. Berk, K.N., *Effective microcomputer statistical software.* American Statistician, 1987. **41**(3): 222-228.
26. Berk, K.N. and I. Francis, *A review of the manuals for BMDP and SPSS.* Journal of the American Statistical Association, 1978. **73**: 65-71.
27. Berndt, E.R., *The Practice of Econometrics: Classic and Contemporary*. 1991, Reading, MA: Addison-Wesley Publishing Company.
28. Bodkin, R.G., *Computation in macroeconometric model-building: Some historical aspects*, in *Advances in Econometrics, Income Distribution, and Scientific Methodology: Essays in Honor of Camilo Dagum*, D.J.Slottje, Editor. 1999, Physica-Verlag: New York. p. 41-60.
29. Bodkin, R.G., L.R. Klein, and K. Marwah, *A History of Macroeconometric Model-Building*. 1991, Brookfield, Vermont: Edward Elgar.
30. Bona, J.L. and M.S. Santos, *On the role of computation in economic theory.* Journal of Economic Theory, 1997. **72**: 241-281.
31. Boschan, C., *The NBER Time Series Data Bank.* Annals of Economic and Social Measurement, 1972. **1**: 193-209.
32. Bracy, D., *PLANETS: Programming Language for the Analysis of Economic Time Series*. 1970, Washington, DC: Social Science Computing Center, Brookings Institution.
33. Bracy, D., *PLANETS: Programming Language for the Analysis of Economic Time Series*. 1972, Washington, DC: Social Science Computing Center, Brookings Institution.
34. Bracy, D., et al., eds. *Computer Programs Available for Distribution*. 1975, Wharton Econometric Forecasting Associates: Philadelphia, PA.
35. Brooks, C., S. Burke, and G. Persand, *Benchmarks and the accuracy of GARCH model estimation.* International Journal of Forecasting, 2001. **17**: 45-56.
36. Brown, J.A.C., H.S. Houthakker, and S.J. Prais, *Electronic computation in economic statistics.* Journal of the American Statistical Association, 1953. **48**(263): 414-428.
37. Brunner, K., *Problems and issues in current econometric practice*. 1972, Columbus,: College of Administrative Science Ohio State University.
38. Bruno, G. and R. De Bonis, *A comparative study of alternative econometric packages wiht an application to Italian deposit interest rates.* Journal of Economic and Social Measurement, 2003. **this issue**.
39. Brynin, M. and R. Smith, *Mapping the Household.* Journal of Economic and Social Measurement, 1994. **20**: 1-17.
40. Byte, *Guide to the IBM Personal Computers.* Byte: The small systems journal, 1984. **9**(9).
41. Byte, *Inside the IBM PCs.* Byte, the small systems journal, 1986. **11**(11).
42. Cartwright, D.W. *The regional economic information system: Data management problems and solutions*. in *Annual Conference of the the Association for University Business and Economic Reseach*. 1979. San Antonio, Texas.
43. Center for Computational Research in Economics and Management Science, *TROLL Primer, 3rd Edition*. 1979, Cambridge, MA: MIT Center for Computational Research in Economics and Management Science.
44. Charemza, W.W. and D.F. Deadman, *New Directions in Econometric Practice*. 1993, Aldershot: Edward Elgar.
45. Cohen, S. and S. Pieper, *The Speakeasy III Reference Manual*. 1979, Chicago: Speakeasy Computing Corporation.
46. Coleman, E.J. and D.W. Cartwright. *Toward a REIS System for the 1980s*. in *Association for University Business and Economic Research Annual Meeting*. 1980. Big Sky, Montana.
47. Condie, J.M., et al., *A brief description of the FRB MODELEASY/FEDEASY econometric language.* Journal of Economic Dynamics and Control, 1983. **5**(1): 75-79.
48. Cooley, T.F. and E.C. Prescott, *Varying parameter regression: a theory and some applications.* Annals of Economic and Social Measurement, 1973. **4**(2): 463-474.
49. Court, R., *The Lucas Critique: did Phillips make a comparable contribution*, in *A.W.H. Phillips: Collected Works in Contemporary Perspective*, R. Leeson, Editor. 2000, Cambridge University Press: Cambridge. p. 460-467.
50. Cribari-Neto, F., *C for Econometricians.* Computational Economics, 1999. **14**(1-2): 135-149.
51. Cripps, T.F. and M. Fetherston, *Cambridge economic policy group methodology*, in *Economic Modelling*, P. Ormerod, Editor. 1979, Heinemann: London. p. 40-52.

52. Cuff, R.N., *On Casual Users.* International Journal of Man-Machine Studies, 1980. **12**:163-187.
53. Cuthbertson, K., S.G. Hall, and M.P. Taylor, *Applied econometric techniques*. 1992, Ann Arbor: University of Michigan Press.
54. Darnell, A.C. and L. Evans, *The limits of econometrics*. 1990, Aldershot, Hants., England: E. Elgar.
55. Data Resources, *EPS Reference Manual*. 1978, Cambridge, MA: Data Resources, Inc.
56. Data Resources, *EPS On-Line: An Example Session*. 1979, Cambridge, MA: Data Resources, Inc.
57. David, M.H., *Managing Panel data for scientific analysis: the role of relational database management systems*, in *Panel Surveys*, D. Kasprzyk, et al., Editors. 1989, John Wiley: New York. p. 226-241.
58. David, M.H. and A. Robbin, *Building New Infrastructures for the Social Science Enterprise: Final Report to the National Science Foundation on the SIPP Access Project, November 1984-December 1991*. 1992, Madison, WI: Institute for Research on Poverty.
59. de Leeuw, F. and E. Gramlich, *The Federal Reserve-MIT Econometric Model.* Federal Reserve Bulletin, 1968: 11-29.
60. de Leeuw, F. and E. Gramlich, *The Channels of Monetary Policy.* Federal Reserve Bulletin, 1969: 472-491.
61. Dempster, A.P., N.M. Laird, and D.B. Rubin, *Maximum likelihood from incomplete data via the EM algorithm.* Journal of the Royal Statistical Society, Series B, 1977. **39**(1): 1-38.
62. Dennis, J.E. and R.E. Welsch, *Techniques for Nonlinear Least Squares and Robust Regression.* Communications in Statistics, 1978. **7**(345–359).
63. Dennis, J.E.J., D.M. Gay, and R.E. Welsch, *An Adaptive Nonlinear Least-Squares Algorithm.* ACM Transactions on Mathematical Software, 1981. **7**: 348–368.
64. Dennis, J.E.J., D.M. Gay, and R.E. Welsch, *Algorithm 573 NL2SOL--An Adaptive Nonlinear Least-Squares Algorithm.* ACM Transactions on Mathematical Software, 1981. **7**: 369-383.
65. Desai, M., D.F. Hendry, and G.E. Mizon, *John Dennis Sargan: An obituary.* Economic Journal, 1997. **107**: 1121-1125.
66. DeWald, W.G., J.G. Thursby, and R.G. Anderson, *Replication in empirical economics: the Journal of Money, Credit and Banking Project.* American Economic Review, 1986. **76**(4): 587-603.
67. Dhrymes, P.J., et al., *Criteria for the evaluation of econometric models.* Annals of Economic and Social Measurement, 1972. **3**(1): 291-324.
68. Dickerson, C.J., ed. *The Software Catalog: Microcomputers. Fall 1983*. 1983, Elsevier: New York.
69. Diebold, F.X., *The Past, Present, and Future of Macroeconomic Forecasting.* Journal of Economic Perspectives, 1998. **12**(2): 175-192.
70. Doan, T., *Email Message*, C.G. Renfro, recipient. 10 March 2003.
71. Don, H. and G.M. Gallo, *Solving large sparse systems of equations in econometric models.* Journal of Forecasting, 1987. **6**: 167-180.
72. Dongarra, J., *Email re The LINPACK Benchmark*, C.G. Renfro, recipient. 22 March 2003.
73. Dongarra, J., P. Luszczek, and A. Petitet, *The LINPACK Benchmark: Past, Present, and Future.* Concurrency: Practice and Experience, 2003. **15**: 803-820.
74. Doornik, J.A., D.F. Hendry, and N. Shephard, *Computationally-intensive econometrics using a distributed matrix-programming language*. 2002.
75. Doornik, J.A. and M. Ooms, *A Package for Estimating, Forecasting, and Simulating AFRIMA Models: Arfima 1.0 for Ox*, in *Discussion Paper, Nuffield College*. 1999: Oxford.
76. Drud, A., *An optimization code for nonlinear econometric models based on sparse matrix techniques and reduced gradients*. 1977, Lyngby: Matematisk Institut Danmarks Tekniske H²jskole. v.
77. Drud, A., *Interfacing Modelling Systems and Solution Algorithms.* Journal of Economic Dynamics and Control, 1983. **5**: 131-149.
78. Drud, A., *A survey of model representations and simulation algorithms in some existing modeling systems.* Journal of Economic Dynamics and Control, 1983. **5**: 5-35.
79. Duesenberry, J.S., et al., *The Brookings quarterly econometric model of the United States*. 1965, Chicago,: Rand McNally.

80. Duesenberry, J.S., et al., *The Brookings Model: Some Further Results*. 1969, Chicago: Rand McNally & Company.
81. Duncan, J.W., *Seeking a Strategy*, in *Report on the Conference on Cataloging and Information Services for Machine-readable Data Files, Arlie House, Wasrrenton, Virginia, March 29-31, 1978*. 1978, Data Use and Access Laboratories: Arlington, VA. p. 118-119.
82. Durbin, J., *Maximum Likelihood Estimation of the Parameters of a System of Simultaneous Regression Equations.* Econometric Theory, 1988. **4**: 159-170.
83. Dynamics Associates, *Economic Forecasting in XSIM User's Guide*. 1979, Waltham, MA: Interactive Data Corporation.
84. Edwards, J.B. and G. Orcutt, *Should aggregation prior to estimation be the rule.* Review of Economics and Statistics, 1969. **51**(4): 409-420.
85. Eisenpress, H., *Forecasting by Generalized Regression Methods. Limited-Information Estimation Procedure IBM 704 Program IB LI*. 1959, New York: International Business Machines Corporation. Data Systems Division.
86. Eisner, M., *TROLL/1 - an interactive computer system for econometric research.* Annals of Economic and Social Measurement, 1972. **1**: 95-96.
87. Eisner, M., *Email re econometric software*, C.G. Renfro, recipient. 27 August 2003.
88. Eisner, M. and R.S. Pindyck, *A generalized approach to estimation as implemented in the TROLL/1 system.* Annals of Economic and Social Measurement, 1973. **2**: 29-51.
89. Ellis, M.E., *Social Science Computing at the University of Wisconsin.* Annals of Economic and Social Measurement, 1972. **1**(2): 237-248.
90. Engle, R.F. and C.W.J. Granger, *Long-run economic relationships : readings in cointegration*. Advanced texts in econometrics. 1991, Oxford ; New York: Oxford University Press.
91. Erdman, L.C., *Recursive Simulation of the Brookings Quarterly Econometric Model of the United States*, in *Master Thesis*. 1966, Massachusetts Institute of Technology: Cambridge, MA.
92. Evans, M.K. and L.R. Klein, *The Wharton Econometric Forecasting Model*. 1967, Philadelphia: Economics Research Unit, University of Pennsylvania.
93. Fair, R.C., *A Short-Run Forecasting Model of the United States Economy*. 1971, Lexington, MA: Lexington Books, D.C.Heath and Company.
94. Fair, R.C., *On the robust estimation of econometric models.* Annals of Economic and Social Measurement, 1974. **3**(4): 667-678.
95. Fair, R.C., *A note on the computation of the Tobit estimator.* Econometrica, 1977. **45**(7): 1723-1727.
96. Fair, R.C., *Specification, estimation, and analysis of macroeconometric models*. 1984, Cambridge, Mass.: Harvard University Press. 479.
97. Fair, R.C., *Specification, estimation, and analysis of macroeconometric models*. Second Edition ed. 1994, Cambridge, Mass.: Harvard University Press.
98. Fair, R.C., *Email re History of econometric software*, C.G. Renfro, recipient. 17 June 2003.
99. Fair, R.C. and W.F. Parke, *The Fair-Parke Program for the Estimation and Analysis of Nonlinear Econometric Models.* 1993: New Haven, CT.
100. Fair, R.C. and J.B. Taylor, *Solution and maximum likelihood estimation of dynamic nonlinear rational expectations models.* Econometrica, 1983. **51**: 1169-1185.
101. Farrell, M.J., *The measurement of productive efficiency.* Journal of the Royal Statistical Society, Series A (General), 1957. **120**(Part III): 253-290.
102. Fay, D., *&Forecast: General Model Simulator User's Manual and Reference Manual*. 1981, Bala Cynwyd, PA: Regional Economics, Chase Econometric Forecasting Associates.
103. Fiorentini, G., G. Calzolari, and L. Panattoni, *Analytic derivatives and the computation of GARCH estimates.* Journal of Applied Econometrics, 1996. **11**: 399-414.
104. Fisher, P.G. and A.J. Hughes Hallett, *Iterative techniques for solving simultaneous equations systems: a view from the economics literature.* Journal of Computational and Applied Mathematics, 1988. **24**: 241-255.
105. Forsund, F.R. and N. Sarafoglou, *On the origins of data envelopment analysis.* Journal of Productivity Analysis, 2002. **17**: 23-40.
106. Forsund, F.R. and N. Sarafoglou, *The Tale of two Research Communities: The Diffusion of Research on Productive Efficiency*. Memorandum No. 08/2003. 2003, Oslo: Department of Economics, University of Oslo.

107. Francis, I., *A comparison of several analysis of variance programs.* Journal of the American Statistical Association, 1973. **68**: 860-865.
108. Francis, I., ed. *Statistical Software: A Comparative Review*. 1981, North Holland: New York.
109. Francis, I., R. Heiberger, and P.F. Velleman, *Criteria and Considerations in the evaluation of statistical program packages.* American Statistician, 1975. **29**: 52-56.
110. Fried, S.S., *Evaluating 8087 performance on the IBM PC.* Byte, 1984. **9**(9): 197-208.
111. Friedman, M. and A.J. Schwartz, *Alternative approaches to analyzing economic data.* American Economic Review, 1991. **81**: 39-49.
112. Friend, I. and P. Taubman, *A short-term forecasting model.* Review of Economics and Statistics, 1964. **46**(3): 229-236.
113. Fromm, G., *Survey of United States Models*, in *The Brookings Model: Perspective and Recent Developments*, G. Fromm and L.R. Klein, Editors. 1975, American Elsevier Publishing Company: New York. p. 376-414.
114. Fromm, G. and G.R. Schink, *Short and long-term simulations with the Brookings Model*, in *Econometric Models of Cyclical Behavior*, B.G. Hickman, Editor. 1972, National Bureau of Economic Research: New York.
115. Fromm, G. and G.R. Schink, *Aggregation and Econometric Models.* International Economic Review, 1973. **16**: 1-32.
116. Fromm, G. and G.R. Schink, *An evaluation of the predictive abilities of a large model: post-sample simulations with the Brookings Model*, in *The Brookings Model: Perspectives and Recent Developments*, G. Fromm and L.R. Klein, Editors. 1975, North Holland: New York.
117. Fromm, G., P. Taubman, and Brookings Institution., *Policy simulations with an econometric model*. 1968, Washington,: Brookings Institution.
118. Gilbert, C.L., *Professor Hendry's econometric methodology.* Oxford Bulletin of Economics and Statistics, 1986. **48**: 283-307.
119. Gilli, M., *Causal Ordering and Beyond.* International Economic Review, 1992. **33**(4): 957-971.
120. Gilli, M., *Computational economic systems : models, methods & econometrics*. Advances in computational economics ; v. 5. 1996, Boston, MA: Kluwer Academic Publishers.
121. Gilli, M. and G. Pauletto, *Krylov methods for solving models with forward-looking variables.* Journal of Economic Dynamics and Control, 1998. **22**: 1275-1289.
122. Goffe, W.L. and R.P. Parks, *The future information infrastructure in economics.* Journal of Economic Perspectives, 1997. **11**(3): 75-94.
123. Goldberger, A.S., *Econometric computation by hand.* Journal of Economic and Social Measurement, 2003(this issue).
124. Goldberger, A.S. and V. Hofer, *A Users Guide to the Multiple Regression Program RGR*. Systems Formulation and Methodology Workshop Paper 6207. 1962, Madison, WI: Social Systems Reserach Institute, University of Wisconsin.
125. Granger, C.W.J., *Modelling economic series : readings in econometric methodology*. 1990, Oxford: Clarendon Press.
126. Granger, C.W.J., *Empirical modeling in economics : specification and evaluation*. 1999, Cambridge ; New York: Cambridge University Press.
127. Green, G.R., *SIMEMOD: A General Purpose Computer Program to Simulate and Forecast with Econometric Models*, in *BEA Staff Paper in Economics and Statistics*. 1970, Bureau of Economic Analysis, US Department of Commerce: Washington, DC.
128. Green, G.R. and I. Green, eds. *The Wharton Short Term Forecasting System Computer Programs*. 1973, Wharton Econometric Forecasting Associates: Philadelphia.
129. Greene, W.H., *Maximum likelihood estimation of econometric frontier functions.* Journal of Econometrics, 1980. **13**: 27-56.
130. Greene, W.H., *On the asymptotic bias of the ordinary least squares estimator of the tobit model.* Econometrica, 1980. **48**: 505-514.
131. Greene, W.H., *Econometric analysis*. 2000, Prentice Hall: Upper Saddle River, N.J.
132. Greene, W.H., *Econometric analysis*. 5th ed. 2003, Upper Saddle River, N.J.: Prentice Hall.
133. Greene, W.H., *Email re EM Algorithm*, C.G. Renfro, recipient. 26 June 2003.
134. Greene, W.H., *Email re LIMDEP characteristics*, C.G. Renfro, recipient. 18 March 2003.
135. Greene, W.H., *Email re Randomly Varying Parameters*, C.G. Renfro, recipient. 17 March 2003.

136. Greene, W.H., *Reply to "a comparative study of alternative econometric packages with an application to Italian deposit rates", by G. Bruno and R. De Bonis.* Journal of Economic and Social Measurement, **this issue**.
137. Griliches, Z., *Productivity, R&D and the data constraint.* American Economic Review, 1994. **84**: 1-23.
138. Hall, B.H., *Email re the History of Econometric Software Development*, C.G. Renfro, recipient. 28 February 2003.
139. Hall, B.H. and R.E. Hall, *Time Series Processor, Version 3.5, User's Manual*. 1980.
140. Hall, R.E., *TSP: Time Series Processor. Source listings of various Fortran routines.* 1968.
141. Hall, R.E., *Email re History of Econometric Software Development*, C.G. Renfro and B.H. Hall, recipients. 27 February 2003: Palo Alto, CA.
142. Hall, S.G., *Email message*, C.G. Renfro, recipient. 18 March 2003.
143. Harris, R.I.D., *Using Cointegration Analysis in Econometric Modeling*. 1995, London: Prentice Hall/Harvester Wheatsheaf.
144. Hausman, J., *An instrumental variable approach to full information estimators for linear and certain nonlinear econometric models.* Economica, 1975. **43**(4): 727-738.
145. Heckman, J.J., *Sample selection bias as a specification error.* Econometrica, 1979. **47**(1): 153-162.
146. Hendry, D.F., *The Estimation of Economic Models With Autoregressive Errors*, Ph.D. dissertation, *Department of Economics, London School of Economics*. 1970, University of London: London.
147. Hendry, D.F., *The structure of simultaneous equation estimators.* Journal of Econometrics, 1976. **4**: 51-88.
148. Hendry, D.F., *Econometrics : alchemy or science? : essays in econometric methodology*. 1993, Oxford, UK ; Cambridge: B. Blackwell. xiv, 518.
149. Hendry, D.F., *Dynamic Econometrics*. 1995, Oxford: Oxford University Press.
150. Hendry, D.F., *Email re Econometric Software on the Microcomputer*, C.G. Renfro, recipient, 16 March 2003.
151. Hendry, D.F., *Email re final version of software survey paper*, C.G. Renfro, recipient. 4 September 2003.
152. Hendry, D.F., *J. Denis Sargan and the origins of the LSE econometric methodology.* Econometric Theory, 2003. **19**: 457-480.
153. Hendry, D.F. and J.A. Doornik, *The impact of computational tools on time-series econometrics*, in *Information Technology and Scholarship*, T. Coppock, Editor. 1999, British Academy: London. p. 257-269.
154. Hendry, D.F. and H.-M. Krolzig, *The Properties of Automatic Gets Modelling*. 2003. p. 20.
155. Hendry, D.F. and F. Srba, *AUTOREG: a computer program library for dynamic econometric models with autoregressive errors.* Journal of Econometrics, 1980. **12**: 85-102.
156. Hendry, D.F. and K.F. Wallis, *Econometrics and quantitative economics*. 1984, Oxford Oxfordshire ; New York, NY: B. Blackwell.
157. Herbert, R.D., *Modelling programming languages--appropriate tools?* Journal of Economic and Social Measurement, 2003. **this issue**.
158. Hirsch, A.A., M. Liebenberg, and G.R. Green, *The BEA Quarterly Econometric Model*, in *BEA Staff Paper*. 1973, Bureau of Economic Analysis, US Department of Commerce: Washington, DC.
159. Holland, P.W. and R.E. Welsch, *Robust Regression Using Iteratively Reweighted Least-Squares.* Communications in Statistics, 1977. **6**: 813–827.
160. Holly, S., B. Rustem, and M.B. Zarrop, eds. *Optimal Control for Econometric Models. An Approach to Economic Policy Formulation*. 1979, St. Martin's Press: New York.
161. Holt, C.C., *Validation and application of macroeconometric models using computer simulation*, in *The Brookings Quarterly Econometric Model of the United States*, J.S. Duesenberry, et al., Editors. 1965, Rand McNally & Company: Chicago. p. 637-650.
162. Holt, C.C., *PROGRAM SIMULATE II, A User's and Programmer's Manual*. 1967, Madison, Wisconsin: Social Systems Research Institute.
163. Holt, C.C., *Email re Econometric Software*, C.G. Renfro, recipient. 18 March 2003.
164. Houthakker, H.S., *Some calculations on electricity consumption in Great Britain.* Journal of the Royal Statistical Society, 1951. **Series A, No. 114, Part III**: 351-371.

165. Hughes Hallett, A.J. and L. Piscitelli, *Simple reordering techniques for expanding the convergence radius of first-order iterative techniques.* Journal of Economic Dynamics and Control, 1998. **22**: 1319-1333.
166. Infometrica, *MOSAIC: A New Tool for Economic Analysis and Forecasting*. 1979, Ottawa: Infometrica Ltd. 6.
167. Intriligator, M.D., R.G. Bodkin, and C. Hsiao, *Econometric models, techniques, and applications*. 2nd ed. 1996, Upper Saddle River, NJ: Prentice Hall.
168. Johnston, J., *Econometrics Retrospect and Prospect.* Economic Journal, 1991. **101**: 51-56.
169. Johnston, J. and J. DiNardo, *Econometric methods*. 4th ed. 1997, New York: McGraw-Hill.
170. Kendrick, D. (ed.), *Research Opportunities in Computational Economics: Report of the NSF Workshop on Research Opportunities in Computational Economics, Washington, DC, March 6, 1991*. 1994: Austin, Texas: Center for Economic Research, Department of Economics.
171. Klein, L.R., *Economic Fluctuations in the United States 1921-1941*. Cowles Commission Monograph. Vol. 11. 1950, New York: John Wiley & Sons, Inc.
172. Klein, L.R., *Single equation vs. equation systems methods of estimation in economics.* Econometrica, 1960. **28**: 866-871.
173. Klein, L.R., *A Textbook of Econometrics*. Second ed. 1974, Englewood Cliffs, NJ: Prentice-Hall.
174. Klein, L.R., *The History of Computation in Econometrics*. 1989. p. 23.
175. Klein, L.R., et al., *An Econometric Model of the United Kingdom*. 1961, Oxford: Basil Blackwell.
176. Klein, L.R. and V. Long, *Capacity Utilization--Concept, Measurement, and Recent Estimates.* Brookings Papers on Economic Activity, 1973. **3**: 743-756.
177. Klein, L.R. and R. Preston, *Some New Results in the Measurement of Capacity Utilization.* American Economic Review, 1967. **57**(1): 34-58.
178. Klein, L.R. and R. Summers, *The Wharton Index of Capacity Utilization*. 1966, Philadelphia, Pennsylvania: Economics Research Unit, Department of Economics, Wharton School of Finance and Commerce, University of Pennsylvania.
179. Knapp, J.L., T.W. Fields, and R.T. Jerome, *A Survey of State and Regional Econometric Models*. 1978, Charlottesville, VA: Tayloe Murphy Institute, University of Virginia.
180. Knuth, D.E., *Von Neumann's first computer program.* ACM Computing Surveys, 1970. **2**(4): 247-260.
181. Koopmans, T.C., *Statistical Inference in Dynamic Economic Models*. First ed. Cowles Commission for Research in Economics. Vol. 10. 1950, New York: John Wiley & Sons. xvi,438.
182. Koopmans, T.C., *On the concept of optimal economic growth*, in *The Econometric Approach to Development Planning*. 1965, North Holland: Amsterdam.
183. Kravis, I.B., *Comparative studies of national incomes and prices.* Journal of Economic Literature, 1984. **22**(1): 1-39.
184. Kravis, I.B., A.W. Heston, and R. Summers, *Real GDP Per Capita for more than one hundred countries.* Economic Journal, 1978. **88**: 215-242.
185. Kravis, I.B. and R.E. Lipsey, *Export prices and the transmission of inflation.* American Economic Review, 1977. **67**(1): 155-163.
186. Kuh, E., *A progress report*, in *The Brookings Model: Some Further Results*, J.S. Duesenberry, et al., Editors. 1969, Rand McNalley & Company: Chicago. p. 3-16.
187. Kuh, E., *Some Notes on the Research Program at the NBER Computer Research Center for Economics and Management Science.* Annals of Economic and Social Measurement, 1972. **1**(2): 233-36.
188. Kuh, E., J. Neese, and P. Hollinger, *Structural Sensitivity in Econometric Models*. 1985, New York: John Wiley & Sons.
189. Kuh, E. and R.E. Welsch, *Econometric Models and Their Assessment for Policy: Some New Diagnostics Applied to the Translog Energy Demand in Manufacturing*, in *Proceedings of the Workshop on Validation and Assessment Issues of Energy Models*, S. Gass, Editor. 1980, National Bureau of Standards: Washington, DC, p. 445-475.
190. Kydland, F.E. and E.C. Prescott, *The econometrics of the general equilibrium approach to business cycles.* Scandinavian Journal of Economics, 1991. **93**(2): 161-178.
191. Kydland, F.E. and E.C. Prescott, *Hours and employment variation in business cycle theory.* Economic Theory, 1991. **1**(1): 63-81.

192. Kydland, F.E. and E.C. Prescott, *The computational experiment: an econometric tool.* Journal of Economic Perspectives, 1996. **10**(1): 69-85.
193. Lane, T., ed. *TROLL Reference Manual (Standard System)*. Second ed. 1979, MIT Center for Computational Research in Economics and Management Science, MIT Information Processing Services: Cambridge, MA.
194. Laurent, S. and J.-P. Peters, *GARCH 2.0: An Ox Package for Estimating and Forecasting Various ARCH Models*. 2001. p. 33.
195. Leamer, E.E., *Specification Searches: Ad Hoc Inference with Non-experimental Data*. 1978, New York: Wiley.
196. Leamer, E.E., *Let's take the con out of econometrics,*. American Economic Review, 1983. **73**: 31-43.
197. Leeson, R., *A.W.H. Phillips M.B.E. (Military Division).* Economic Journal, 1994. **104**(May 1994): 605-618.
198. Leeson, R., ed. *A.W. H. Phillips: Collected Works in Contemporary Perspective*. 2000, Cambridge University Press: Cambridge.
199. Leontief, W., *Structural matrices of national economies.* Econometrica, 1949. **17**(Issue Supplement: Report of the Washington Meeting): 273-282.
200. Leontief, W.W., *Computational problems arising in connection with economic analysis of industrial relationships*, in *Proceedings of a Symposium on Large-Scale Digital Calculating Machinery*. 1948, Harvard University Press: Cambridge, MA.
201. Leontief, W.W., *The Structure of the American Economy*. 1949, New York: Oxford University Press.
202. Lilien, D.M., *MicroTSP and Eviews*, C.G. Renfro, recipient. 2003.
203. Ling, R.F., *General considerations in the design of an interactive system for data analysis.* Communications of the Association for Computing Machinery, 1980. **23**: 147-154.
204. Liu, T.-c., *Underidentification, structural estimation and forecasting.* Econometrica, 1960. **28**: 855-865.
205. Liu, T.-c., *Survey of United States models: discussion*, in *The Brookings Model: Perspective and Recent Developments*, G. Fromm and L.R. Klein, Editors. 1975, American Elsevier Publishing Company: New York. p. 415-418.
206. Longley, J.W., *An appraisal of least squares programs for the electronic computer from the point of view of the user.* Journal of the American Statistical Association, 1967. **62**: 819-841.
207. Lucas, R.E., Jr, *Econometric policy evaluation: a critique*, in *The Phillips Curve and Labor Markets*, K. Brunner and A.H. Meltzer, Editors. 1976, North Holland: Amsterdam.
208. Mackie-Mason, J.K., *Econometric software: a user's view.* Journal of Economic Perspectives, 1992. **6**: 165-188.
209. MacKinnon, J.G., *Model specification tests and artificial regressions.* Journal of Economic Literature, 1992. **30**: 102-146.
210. Maunder, W.F. and M.C. Fleming, *Reviews of United Kingdom statistical sources.* Journal of Economic and Social Measurement, 1986. **14**: 271-276.
211. McCarthy, M.D., *The Wharton Quarterly Econometric Forecasting Model Mark III*. 1973, Philadelphia, PA: Economic Research Unit, Department of Economics, Wharton School of Finance and Commerce.
212. McCarthy, M.D., *The Cowles Commission, the Brookings Project, and the econometric services industry: successes and possible new directions: a personal view.* Econometric Theory, 1992. **8**: 383-401.
213. McCracken, M.C., *A computer system for econometric research.* Social Science Information, 1967. **5**: 151-158.
214. McCracken, M.C. *Data administration in an information system.* in *Conference on Government Information Systems*. 1967. Ottawa: Economic Council of Canada.
215. McCracken, M.C. and K. May, *The Development of MOSAIC.* Journal of Economic and Social Measurement, 2003. **this issue**.
216. McCracken, M.C. and C.A. Sonnen, *A system for large econometric models: management, estimation, and simulation*, in *Proceedings of the Association for Computing Machinery Annual Conference, August 1972*. 1972, Association for Computing Machinery.

217. McCullough, B.D., *A review of RATS v4.2: Benchmarking numeric accuracy.* Journal of Applied Econometrics, 1997. **12**: 181-190.
218. McCullough, B.D., *Assessing the reliability of statitistical software: Part I.* American Statistician, 1998. **52**(4): 358-366.
219. McCullough, B.D., *Assessing the reliability of statistical software: Part II.* American Statistician, 1999. **53**(2): 149-159.
220. McCullough, B.D., *Econometric software reliability: Eviews, LimDep, Shazam, and TSP.* Journal of Applied Econometrics, 1999. **14**(2): 191-202.
221. McCullough, B.D., *The accuracy of Mathematica 4.0 as a Statistical Package.* Computational Statistics, 2000. **15**: 279-299.
222. McCullough, B.D. *Experience with the StRD: Application and Interpretation*. in *Proceedings of the 31st Symposium on the Interface: Models, Prediction, and Computing*. 2000. Fairfax, Virginia: Interface Foundation of North America.
223. McCullough, B.D., *Is it safe to assume that software is accurate?* International Journal of Forecasting, 2000. **16**: 349-357.
224. McCullough, B.D., *Some details of nonlinear estimation*, in *Numerical Methods in Statistical Computing for the Social Sciences*, M. Altman, J. Gill, and M. McDonald, Editors. 2003, Wiley: New York.
225. McCullough, B.D., *Wilkinson's tests and econometric software.* Journal of Economic and Social Measurement, 2003. **this issue**.
226. McCullough, B.D. and C.G. Renfro, *Benchmarks and software standards: a case study of GARCH procedures.* Journal of Economic and Social Measurement, 1998. **25**: 59-71.
227. McCullough, B.D. and C.G. Renfro, *Some numerical aspects of nonlinear estimation.* Journal of Economic and Social Measurement, 2000. **26**: 63-77.
228. McCullough, B.D. and H.D. Vinod, *The numerical reliability of econometric software.* Journal of Economic Literature, 1999. **37**(2): 633-665.
229. McCullough, B.D. and H.D. Vinod, *Verifying the solution from a nonlinear solver: a case study.* American Economic Review, 2003. **93**(3): 873-892.
230. McCullough, B.D. and B. Wilson, *On the accuracy of statistical procedures in Microsoft Excel 97.* Computational Statistics & Data Analysis, 1999. **31**: 27-37.
231. McCullough, B.D. and B. Wilson, *On the accuracy of statistical procedures in Microsoft Excel 2000 and Excel XP.* Computational Statistics & Data Analysis, 2002. **40**: 713-721.
232. McGuckin, R.H. and S.V. Nguyen, *Public use microdata: disclosure and usefulness.* Journal of Economic and Social Measurement, 1990. **16**: 19-40.
233. Mendelssohn, R.C. *The BLS computer language for quantitative economic research*. in *Conference on Government Information Systems, October 5-6, 1967*. 1967. Ottawa, Canada: Economic Council of Canada.
234. Mendelssohn, R.C., *The New on-line BLS database and information system*, in *Readings in Business and Economic Research Management: Execution and Enterprise*, J.D. Fischer, Editor. 1980, Office of Research Administration, School of Business, University of Wisconsin: Madison, WI. p. 86-89.
235. Mills, T.C., *The Econometric Modeling of Financial Time Series*. 1993, Cambridge: Cambridge University Press.
236. Mills, T.C., *The Econometric Modeling of Financial Time Series*. Second ed. 1999, Cambridge: Cambridge University Press.
237. Morgan, M.S., *The history of econometric ideas*. Historical perspectives on modern economics. 1990, Cambridge England ; New York: Cambridge University Press. xv, 296.
238. Morgenstern, O., *On the Accuracy of Economic Observations*. 1960, Princeton, NJ: Princeton University Press.
239. Nagar, A.L., *Stochastic Simulation of the Brookings Econometric Model*, in *The Brookings Model: Some Further Results*, J.S. Duesenberry, et al., Editors. 1969, Rand McNally & Company: Chicago. p. 425-456.
240. Nepomiastchy, P. and A. Ravelli, *Adapted methods for solving and optimizing quasi-triangular econometric models.* Annals of Economic and Social Measurement, 1978. **6**(5): 555-562.
241. Nerlove, M., *A tabular survey of macroeconometric models.* International Economic Review, 1966. **7**: 127-175.

242. Nerlove, M., *History of panel data econometrics 1861-1997*, in *Essays in Panel Data Econometrics*. 2002, Cambridge University Press: Cambridge.
243. Nerlove, M., *Email re EM algorithm*, C.G. Renfro, recipient. 26 June 2003.
244. Norman, A.L., L.S. Lasdon, and J.K. Hsin, *A comparison of methods for solving and optimizing a large non-linear econometric model.* Journal of Economic Dynamics and Control, 1983. **6**(1/2): 3-24.
245. Norman, M., *Multiple Regression Program (AUTO: functional description and instructions for use)*, in *The Wharton Short Term Forecasting System Computer Programs*, G.R. Green and I. Green, Editors. 1973, Wharton Econometric Forecasting Associates: Philadelphia, PA. p. 1-26.
246. Norman, M., H. Shapiro, and R. Rasche, *The SIM Model Solution Program*. 1967, Cambridge, MA: MIT Center for Computational Research in Economics and Management Science, MIT Information Processing Services.
247. Orcutt, G., *Toward partial redirection of econometrics.* Review of Economics and Statistics, 1952. **34**: 195-213.
248. Orcutt, G., S. Caldwell, and R. Wertheimer, *Policy Exploration Through Microanalytic Simulation*. 1976, Washington, DC: Urban Institute.
249. Pauletto, G., *Computational solution of large-scale macroeconometric models*. Advances in computational economics ; v. 7. 1997, Boston: Kluwer Academic Publishers. xiv, 161.
250. PC Age, *IBM takes a spin on the hard disk: Big Blue announces a hard disk PC and new DOS expansion*, in *Personal Computer Age*. 1983. p. 94-95.
251. Pease, J. and R. Lepage, *An evaluation of selected microcomputer statistical programs, Working Paper 15*. 1984: Department of Agricultural Economics, Michigan State University.
252. Pesaran, M.H. and G.C. Harcourt, *Life and Work of John Richard Nicholas Stone 1913-1991.* The Economic Journal, 2000. **110**: F146-F165.
253. Pesaran, M.H. and B. Pesaran, *Microfit. A Interactive Econometric Software Package. User Manual*. 1987, Oxford: Oxford University Press.
254. Pesaran, M.H. and B. Pesaran, *Microfit 3.0. An Interactive Econometric Software Package. User Manual*. 1991, Oxford: Oxford University Press.
255. Pesaran, M.H. and L.J. Slater, *Dynamic Regression: Theory and Algorithms*. 1980, Chichester: Ellis Horwood.
256. Pesaran, M.H. and M. Wickens, *Handbook of applied econometrics*. Blackwell handbooks in economics. 1995, Oxford, UK ; Cambridge, Mass.: Blackwell. v. <1 >.
257. Peterson, W., *Computer software for a large econometric model*, in *The Cambridge Multisectoral Dynamic Model of the British Economy*, T.S. Barker and W. Peterson, Editors. 1987, Cambridge University Press: Cambridge. p. 105-121.
258. Peterson, W., *Econometric Computing in Cambridge*, C.G. Renfro, recipient. 2003.
259. Peterson, W., T. barker, and R. van der Ploeg, *Software support for multisectoral dynamic models of national economies.* Journal of Economic Dynamics and Control, 1983. **5**(1): 81-108.
260. Peterson, W., T.S. Barker, and L.A. Winters, *IDIOM, an International Dynamic Input-Output Model*, in *Proceedings of the 7th International Input-Output Conference, 1979*. 1984, UNIDO: New York.
261. Phillips, P.C.B., *The Structural Estimation of a Stochastic Differential Equation System.* Econometrica, 1972. **40**(6): 1021-1041.
262. Phillips, P.C.B., *Email re the History of Econometric Software*, C.G. Renfro, recipient. 28 March 2003.
263. Phillips, P.C.B. and V. Hall, *Early development of econometric software at the University of Auckland.* Journal of Economic and Social Measurement, 2003(this issue).
264. Pilozzi, G., *A new approach to the dissemination of CANSIM data.* Canadian Statistical Review, 1976. **51**: 5-8.
265. Post, E., *Real programmers don't use Pascal.* Datamation, 1983. **29**(7).
266. Prais, S.J., *The estimation of equivalent-adult scales from family budgets.* Economic Journal, 1953. **63**(252): 791-810.
267. Prais, S.J. and H.S. Houthakker, *The Analysis of Family Budgets*. 1955, Cambridge: Cambridge University Press.
268. Prescott, E.C. and T.F. Cooley, *Estimation in the Presence of Stochastic Parameter Variation.* Econometrica, 1976. **44**(1): 167-184.

269. Qin, D., *The formation of econometrics : a historical perspective*. 1993, Oxford, England: Clarendon Press. xii, 212.
270. Quandt, R.E., *Computational Problems and Methods*, in *Handbook of Econometrics*, Z. Griliches and M.D. Intriligator, Editors. 1983, North Holland: Amsterdam. p. 699-764.
271. Raduchel, W. and O. Eckstein, *Economic Modeling Languages: the DRI Experience.* Journal of Economic Dynamics and Control, 1983. **5**(1): 59-74.
272. Ramsey, F.P., *A mathematical theory of savings.* Economic Journal, 1928. **38**: 543-559.
273. Reisman, A., et al., *On the voids in U.S. national educational statistics.* Journal of Economic and Social Measurement, 1986. **14**: 357-366.
274. Renfro, C.G., *An interactive regression program for the PDP-10. Description and User Manual*. 1970, Washington, DC: Brookings Quarterly Model Project, Brookings Institution.
275. Renfro, C.G., *On the development of a comprehensive public data base for aggregate state and local economic data.* Review of Public Data Use, 1979. **7**(5/6): 1-10.
276. Renfro, C.G., *Economic Data Base Systems: Some reflections on the state of the art.* Review of Public Data Use, 1980. **8**(3): 121-140.
277. Renfro, C.G., *An on-line information system for aggregate state and local area economic data.* Journal of the American Society for Information Science, 1980. **31**(5).
278. Renfro, C.G., *Reflections on a theme by Mendelssohn: the LABSTAT data base system*, in *Readings in Business and Economic Research Management: Execution and Enterprise*, J.D. Fischer, Editor. 1980, Office of Research Administration, School of Business, University of Wisconsin: Madison, WI. p. 86-90.
279. Renfro, C.G., *The Computer as a Factor of Production in Economic Data Provision and Research*, in *American Economic Association Annual Meeting*. 1981: Washington, DC.
280. Renfro, C.G. *The design of an information system for state and local government planning and analysis*. in *Proceedings of the Eighth European Urban Data Management Symposium*. 1981. Oslo, Norway: Norwegian Ministry of the Environment.
281. Renfro, C.G., *On the Development of Econometric Modeling Languages: MODLER and its first twenty-five years.* Journal of Economic and Social Measurement, 1996. **22**(4): 241-311.
282. Renfro, C.G., *Economic Data Base Systems: Further reflections on the state of the art.* Journal of Economic and Social Measurement, 1997. **23**(1): 43-85.
283. Renfro, C.G., *Normative Considerations in the development of a software package for econometric estimation.* Journal of Economic and Social Measurement, 1997. **23**(4): 277-330.
284. Renfro, C.G., *Macroeconometric models and changes in measurement concepts: an overview.* Journal of Economic and Social Measurement, 1998. **24**(2): 63-82.
285. Renfro, C.G. (editor), *A compendium of existing econometric software packages.* Journal of Economic and Social Measurement, 2003. **this issue**.
286. Renfro, C.G., *The early development of econometric modeling languages.* Journal of Economic and Social Measurement, 2003. **this issue**.
287. Renfro, C.G. and P.A. Coomes, *The Kentucky Economic Information System User Manual*. 1976, Lexington, Kentucky: Office for Research, College of Business and Economics, University of Kentucky.
288. Rosen, S., *Electronic Computers: A historical survey.* ACM Computing Surveys, 1969. **1**(1): 7-36.
289. Rubin, D.B., *Inference and missing data.* Biometrika, 1976. **63**(3): 581-592.
290. Sadowsky, G., *MASH: A Computer System for Microanalytic Simulation for Policy Exploration*. 1977, Washington, DC: Urban Institute.
291. Sall, J.P., *SAS Econometric and Time-series Library. Preliminary Test Version for SAS 79.2*, in *SAS Institute Technical Report*. 1979, SAS Institute, Inc.: Raleigh, North Carolina.
292. Samuelson, P.A., *The art and science of macromodels over 50 years*, in *The Brookings Model: Perspective and Recent Developments*, G. Fromm and L.R. Klein, Editors. 1975, North-Holland: Amsterdam. p. 3-10.
293. Sargan, J.D., *Wages and prices in the United Kingdom: a study in econometric methodology*, in *Econometric Analysis for National Economic Planning*, P.E. Hart, G. Mills, and J.K. Whitaker, Editors. 1964, Butterworths: London. p. 25-63.

294. Sargent, T.J. and C.A. Sims, *Business cycle modeling without pretending to have too much a priori economic theory*, in *New Methods of Business Cycle Reserach*. 1977, Federal Reserve Bank of Minneapolis: Minneapolis, MN. p. 45-109.
295. Sarnak, N. and E. Jaffe, *8087 performance considerations.* PCTech Journal, 1983. **1**(2): 30-47.
296. SAS, *SAS/ETS user's guide, version 5 edition*. 1984, Cary, N.C.: SAS Institute.
297. SAS, *SAS/ETS user's guide, version 6*. 2nd ed. 1993, Cary, N.C.: SAS Institute.
298. SAS, *SAS/ETS user's guide, version 8*. 1999, Cary, NC: SAS Pub.
299. Schink, G.R., *Simulation with Large Econometric Models. Paper presented to the Association for Computing Machinery meetings, Denver, Colorado, 12 June 1970.* 1970.
300. Schink, G.R., *The Brookings Quarterly Model: An aid to longer term economic policy analysis*, in *Econometric Model Performance: Comparative Simulation Studies of the U.S. Economy*, L.R. Klein and E. Burmeister, Editors. 1976, University of Pennsylvania Press: Philadelphia, PA. p. 307-321.
301. Senko, M.E., *Information systems: records, relations, sets, entities, and things.* Information Systems, 1975. **1**: 3-13.
302. Shiskin, J., A.H. Young, and J.C. Musgrave, *The X-11 Variant of Census Method II Seasonal Adjustment, Technical Paper No. 15*. 1967, Washington, DC: Bureau of the Census, U.S. Department of Commerce.
303. Siegel, J., *Moving data between PCs and mainframes*, in *Byte Guide to the IBM PC*. 1984. p. 248-255.
304. Sims, C.A., *Macroeconomics and reality.* Econometrica, 1980. **48**: 1-48.
305. Slater, L.J., *Regression analysis.* Computer Journal, 1962: 1-5.
306. Slater, L.J., *Fortran Programs for Economists. Department of Applied Economics Occasional Paper 13*. 1967, Cambridge: Cambridge University Press.
307. Slater, L.J., *More Fortran Programs for Economists. DAE Occasional Paper 34*. 1972, Cambridge: Cambridge University Press.
308. Slater, L.J., *GEM: A General Econometric Matrix Program. DAE Occasional Paper 46*. 1976, Cambridge: Cambridge University Press.
309. Slater, L.J., *Email re Air Conditioning*, C.G. Renfro, recipient. 1 July 2003.
310. Slater, L.J., *Email re EDSAC and the DAE*, C.G. Renfro, recipient. 2 May 2003.
311. Slater, L.J., *Email re Your paper and follow on*, C.G. Renfro, recipient. 10 June 2003, 3:22PM.
312. Slater, L.J., *Email Re: Re: Your paper and follow on*, C.G. Renfro, recipient. 10 June 2003, 3:28PM.
313. Slater, L.J., *Recollections on Early Days in the Cambridge Computing Laboratory.* Journal of Economic and Social Measurement, **this issue**.
314. Slater, L.J. and T.S. Barker, *Regression Program for Titan*. 1967, Cambridge: Department of Applied Economics, University of Cambridge.
315. Slater, L.J. and T.S. Barker, *REX Regression Program for Titan*. 1971, Cambridge: Department of Applied Economics, University of Cambridge.
316. Smith, B., *Showtime in the City.* DTP Desktop Publishing: Professional Presentation Solutions, 1988: 22-23.
317. Startz, R., *8087: Applications and Programming for the IBM PC and Other PCs*. 1983, Bowie, MD: Robert J. Brady Company.
318. Stokes, H.H., *Email re Bayesian Program, etc.*, C.G. Renfro, recipient. 2003.
319. Stokes, H.H., *The evolution of econometric software design: A developer's view.* Journal of Economic and Social Measurement, 2003. **this issue**.
320. Stokes, H.H., *On the advantage of using two or more econometric software systems to solve the same problem.* Journal of Economic and Social Measurement, 2003. **this issue**.
321. Stone, R., J. Aitchison, and A.J.C. Brown, *Some Estimation Problems in Demand Analysis.* The Incorporated Statistician, 1955. **5**(4): 1-13.
322. Stone, R. and A. Brown, *A Computable Model of Economic Growth*. A Program for Growth, ed. R. Stone. Vol. 1. 1962, Cambridge: Department of Applied Economics.
323. Stroud, A., *The Multiple Regression Program RGR*. 1961, Madison, WI: Social Systems Research Institute, University of Wisconsin.
324. Suits, D.B., *The theory and application of econometric models*. 1963, Athens,: Center of Economic Research. xiii, 147.

325. Suits, D.B., *Email re Solution of the Klein-Goldberger model and early software used at the University of Michigan*, C.G. Renfro, recipient. 28 March 2003.
326. Surrey, M.J.C., *The Analysis and Forecasting of the British Economy*. 1971, Cambridge: Cambridge University Press.
327. Taeuber, R.C. and R.C. Rockwell, *National social data series: a compendium of brief descriptions.* Review of Public Data Use, 1982. **10**(1-2): 23-111.
328. Triandafillou, P.N., *CANSIM/SIS--A new phase of development.* Canadian Statistical Review, 1975. **50**: 4-6.
329. Uebe, G., G. Huber, and J. Fischer, *Macro-Econometric Models - An International Bibliography*. 1988, Aldershot: Gower.
330. Velleman, P.F. and R.E. Welsch, *On evaluating interactive statistical program packages.* Communications in Statistics, Part B--Simulation and Computation, 1976. **5**: 197-208.
331. Velleman, P.F. and M.A. Ypellar, *Constructing regressions with controlled features: a method for probing regression performance.* Journal of the American Statistical Association, 1980. **75**: 839-844.
332. Wallis, K.F., et al., *Models of the UK Economy*. 1984, Oxford: Oxford University Press.
333. Wallis, K.F., et al., *Models of the UK Economy: A Second Review by the ESRC Macroeconomic Modelling Bureau*. 1985, Oxford: Oxford University Press.
334. Wallis, K.F., et al., *Models of the UK Economy: A Third Review by the ESRC Macroeconomic Modelling Bureau*. 1986, Oxford: Oxford University Press.
335. Wallis, K.F., et al., *Models of the UK Economy: A Fourth Review by the ESRC Macroeconomic Modelling Bureau*. 1987, Oxford: Oxford University Press.
336. Wasserman, A.I., *Information system design methodology.* Journal of the American Society for Information Science, 1980. **31**: 5-24.
337. Wescott, R.F. and M.V. Yates, *The PC Corner: desktop publishing is for economists.* Business Economics, 1989. **24**: 51-53.
338. Wetherill, G.B., et al., *The design and evaluation of statistical software for microcomputers.* The Statistician, 1985. **34**: 391-427.
339. White, H., R.F. Engle, and C.W.J. Granger, *Cointegration, causality, and forecasting : a festschrift in honour of Clive W.J. Granger*. 1999, Oxford ; New York: Oxford University Press. vi, 497.
340. White, K.J., *A General Computer for Econometric Models--Shazam.* Econometrica, 1978. **46**(1): 239-240.
341. White, K.J., *Email*, C.G. Renfro, recipient. 11 March 2003.
342. Wickens, M., *Email re early computing at the LSE*, C.G. Renfro, recipient. 24 April 2003.
343. Wiggins, V., *Email re Interface versus organization*, C.G. Renfro, recipient. 9 April 2003.
344. Wiggins, V., *Email re Stata entry for Compendium for econometric software*, C.G. Renfro, recipient. 1 April 2003.
345. Wilkes, M.V. and W. Renwick, *The EDSAC, an electronic calculating machine.* Journal of Scientific Instruments, 1949. **26**: 385-391.
346. Wilkes, M.V., D.J. Wheeler, and S. Gill, *The Preparation of Programs for an Electronic Digital Computer: With special reference to the EDSAC and the use of a library of subroutines*. 1951, Cambridge, MA: Addison Wesley.
347. Young, R.M., *Email re the history of econometric software*, C.G. Renfro, recipient. 25 March 2003.
348. Zafft, W., G. Solomon, and R. Perron, *TROLL Reference Manual, in installments*. 1973-1976, New York: National Bureau of Economic Research.
349. Zellner, A., *PCBRAP: A PC Bayesian Regression Program.* American Statistician, 1989. **43**(1): 124.
350. Zellner, A., *Email re History of Econometric Software Development*, C.G. Renfro, recipient. 27 February 2003.
351. Zellner, A., A. Stroud, and L.C. Chau, *Program for Computing Seemingly Unrelated Regressions Estimates*. 1963, Madison, Wisconsin: Social Systems Research Institute, Department of Economics.

352. Zellner, A., A. Stroud, and L.C. Chau, *Program for Computing Two and Three Stage Least Squares and Associated Statistics*. 1963, Madison, Wisconsin: Social Systems Research Institute, Department of Economics.
353. Zivot, E. and J. Wang, *Modelling Financial Time Series with S-PLUS*. May 22, 2002 ed. 2002, Seatle, WA: Insightful Corporation.