1

Econometric software

Charles G. Renfro

1.1 Introduction

The production and use of econometric software began at the University of Cambridge in the early 1950s as a consequence of the availability there of the first operative stored-program electronic computer, the EDSAC (Electronic Delay Storage Automatic Calculator). This machine became available for academic research starting in or about 1951 [214, 248, 249]. Among the first to use it were members of the Department of Applied Economics (DAE), some of whom had been invited to Cambridge by the Director, Richard Stone. From the beginning, they employed the EDSAC to produce work that remains well regarded [8, 77, 124, 195]. They also appear to have been the first to describe in print the workaday process of using a stored-program computer [36]. There, Lucy Slater, working with Michael Farrell, wrote the first distinguishable econometric software package, a regression program [14, 224, 225]. However, these economists were not alone in their early adoption of emerging computing technologies, for previously, in 1947, at one of the first gatherings of computer designers, the Symposium on Large-Scale Digital Calculating Machinery, Wassily Leontief [153] had presented some of his then current work on inter-industry relationships to provide an example of a challenging computational problem. He used the existing electromechanical 'Mark I' Automatic Sequence Control Calculator at Harvard, which, although not actually a stored-program electronic computer, can nonetheless be described as the first 'automatic digital computer' [248]. The Mark I also represents IBM's start in a business it was later to dominate.

At the time, worldwide, these machines were the only automatic computing devices generally available for research. Consequently, other computationally inclined

Handbook of Computational Econometrics edited by D. A. Belsley, E. J. Kontoghiorghes

economists, including Lawrence Klein, Arthur Goldberger, and their colleagues at the University of Michigan [87], still used electromechanical desktop calculators. However, in 1954, the semi-electronic IBM Card Programmed Calculator (CPC) became available to them and was first used to estimate moments in preparation for the estimation of the parameters of the Klein–Goldberger model [138]. A few years later, in 1958–59, at what would become the Lawrence Livermore Laboratory, Frank and Irma Adelman were the first to solve an econometric model using a computer, employing an IBM 650 [6, 7, 139]. Contemporaneously, possibly at the IBM Scientific Center in New York, Harry Eisenpress wrote the first program to perform limited information maximum likelihood estimation [60]. A few years earlier, with Julius Shiskin in Washington, DC [218], he had created the Census X-11 seasonal adjustment method using a UNIVAC (UNIVersal Automatic Computer), a near relative to the EDSAC [214]. This UNIVAC, first installed at the US Bureau of the Census in 1951, was the first stored-program computer to be sold commercially.

This description of 1950s computer use by economists possibly reads like a series of selectively chosen exhibits, but actually it appears to be the complete early record, with the exception of work by others at the DAE [14, 15, 44, 195] and the start of Robert Summers' Monte Carlo study of estimator properties [235]. Throughout the 1950s, computers were scarce. They were also expensive to use – an hour of machine time could cost literally triple the monthly salary of the economist using it [7]. It was only during the next two decades, starting in the early 1960s, that they began to proliferate and economists more commonly became users [30, 52]. A number of econometric firsts occurred during the 1960s, including the implementation of a variety of increasingly computationally complex techniques, among them two- and three-stage least squares, seemingly unrelated regression equations, and full information maximum likelihood [204]. In addition, beginning in about 1964, economists created some of the earliest large-scale computer databases, together with the software to manage them [160–162]. However, progress was neither uniform nor universal: even in the mid-1960s, the Wharton model, a direct descendent of the Klein–Goldberger model [31, 127], was still being solved by using an electromechanical desktop calculator [203, 216]. It was only during the next few years that economists at the University of Pennsylvania first used an electronic computer to solve macroeconometric models as nonlinear simultaneous equation systems [52, 68, 69, 216].

As indicated, the level of sophistication progressively increased during this decade. Much of this work represents the efforts of individuals, although frequently in the context of formal research projects [57, 58, 68, 69, 159, 197]. It typically began with the creation of single purpose programs to estimate parameters, manage datasets, and later solve macroeconometric models. However, towards the end of the 1960s, with the advent of time-sharing computers, the first network-resident interactive econometric software systems began to be created [203, 207], an important step in the development of the modern econometric computational environment. In the early 1970s came attempts to focus this research: in particular, at the MIT Center for Computational Research in Economics and Management Science, economists began to devote considerable effort to the study of relevant computer algorithms [29, 48–50, 120, 147–150] and closely related regression diagnostics [19, 21, 22]. Following these advances, the 1970s saw the development of wide-area online telecommunications-linked economic database, analysis, and econometric modeling systems, which, by the end of the decade, became used worldwide [3, 4,

56, 205, 206]. As a result, at the beginning of the 1980s, economists were ready to adopt the then emerging microcomputer and, by the middle of that decade, had begun to use it as the primary locus for analytical processing, including by 1985 the solution of econometric models of 600 and more equations [209]. However, it was only in late 1984 that it began to become common for econometric models to be mounted on more than a single computer [203].

Building upon these foundations laid in the 1960s and early 1970s, economists have since created not only sophisticated, targeted applications but also specialized econometric modeling and programming languages, as well as generally applicable econometric software for parameter estimation and hypothesis testing that can be used with time-series, cross-section, and panel data. At each stage, this work has affected the disciplinary progress of economics and econometrics. Simultaneously, some of this work fed back into the more general development of software, including operating systems, spreadsheet packages (such as VisiCalc and Lotus 1-2-3 specifically), and a variety of other applications [204]. Last, but certainly not least, because of the subject matter of economics, econometric software has played a critical part in the management of the performance of the world's economies, in its effect both on the timely dissemination of economic information and on the development of the modern tools of applied economics. Of all disciplines, economics has been one of those most affected by the use of the electronic computer, and in turn, in no small measure, this use has helped to enable the broadly consistent expansion and development of both industries and economies.

From the first availability of the electronic computer, economists were obviously primed to become users, raising the question: Why? A reflective comment made by Klein in 1950 suggests a reason [134, p. 12]. Referring to the seminal Cowles Commission work and the methodological progress of the 1940s – and seemingly unaware of what would occur within the next decade – he rather pessimistically observed that:

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

Were it not for such preserved commentary, it might now be difficult to imagine that it appeared at least to some economists at the beginning of the 1950s that econometric methodology faced a serious computational roadblock. The sense of inevitability that hindsight provides makes it easy to overlook the importance of the timing of the adoption of the electronic computer by economists. The distance between that time and ours is accentuated by the watershed effect of the creation of ARPANET, the precursor network to the Internet, in about 1970 and the equally potent consequence of the introduction of the first microcomputers in the mid-1970s.

One of the most significant differences between then and now is that, in those earlier times, especially in the years prior to 1975, it was almost requisite for any computer user to begin by writing some code, and this was certainly true in those instances in which an application might involve new techniques. Then there was so much a need to

program, and comparatively so much less computer use than today, that practically any econometric software written prior to 1975 involves some aspect that can be declared a 'first,' or at minimum viewed as involving some type of pioneering effort. Although there were certainly instances of shared software use by (comparatively small groups of) economists during the period between 1960 and 1980 [33, 35, 89, 115, 131, 160, 196, 222, 223], it was usual nevertheless that an intending hands-on user generally needed to learn to program, at least to some degree, but often at a rather fundamental level. Until at least the mid-1970s, an applied economist wishing to employ a computer would commonly either need to start from scratch, or, at best, be provided a box of cards (or, sometimes, a paper tape), onto which were punched the source code statements for a program – even if by 1975, perhaps earlier, it was not unusual for the 'box of cards' to have been replaced by a machine-readable card image file. Specific code changes were often necessary, at minimum to permit data transformations. For calculations associated with a particular application, a substantial amount of new coding might be required. Today, in contrast, it is only someone operating at the developmental leading edge of econometrics who may need to be proficient at programming, but even then not always in the use of a computer language such as Assembly, C++, C#, or Fortran, or in such a way that might require understanding of the computer's operating system and specific hardware protocols.

However, to declare the beginning of a new epoch is not to say that all the heavy lifting has been done. The year of the film *2001: A Space Odyssey* has come and gone, but we do not yet converse casually with articulate, seemingly intelligent computers, like HAL, nor simply declare in broad outline, via the keyboard or some other interface, what we wish to do and then have it done, automatically, effortlessly, and appropriately. In important respects, the computer still remains a new technology. To present a reliable assessment of the current state of the art, it is therefore necessary to proceed carefully, as well as to begin by establishing a proper framework, so as to be able to describe and evaluate current progress. For instance, there is a useful distinction to be made between creating software designed to perform a specific task versus developing a program for a more generally defined purpose, possibly intended for use by others. A routine might be written specifically to calculate and display a set of parameter estimates. Alternatively, the goal might be to create a program to be used to build a 'model,' or a class of models, or some other composite, possibly quite complex, undertaking. The first of these cases involves a pure computational problem, the success of which might be assessed by the degree to which the individual calculations are accurately, efficiently, and even elegantly or perhaps quickly performed. This qualitative evaluation can also include a determination of the degree to which the results are informatively and even attractively displayed. In the second, more general, case, what is required is potentially much more elaborate computer code to perform an integrated series of tasks, not all of which are purely computational problems. These tasks might also incorporate operations not likely to be immediately classified as either economically or econometrically relevant. Such operations, which can include the storage, retrieval, and management of data, as well as the development of the human interface of a program, are nevertheless as much a part of econometric software creation as is the programming of an estimator. Furthermore, whether or not these elements are individually considered to be econometrically interesting, the future development of economics and econometrics may depend upon a successful solution to the problem each of them poses.

A perspective on the underlying issues can be obtained by considering the usage of the contrasting terms, 'computational econometrics' and 'econometric computing.' Achim Zeileis [252, p. 2988] has recently suggested that the first of these is topically 'mainly about [mathematical or statistical] methods that require substantial computations' whereas the second involves the algorithmic translation of 'econometric ideas into software.' Initially, economists began to use the computer as a means to an end, simply to make necessary calculations more rapidly than could be done manually, without a great deal of introspection. This lack of introspection has lingered. A general appreciation that the execution of these computations can have disciplinarily significant implications has been slow to emerge: 'simply computational' remains a common wave-of-the-hand expression used to suggest the existence of only a few remaining, possibly laborious, but essentially straightforward demonstrative steps. Perhaps as a consequence, the self-conscious consideration of software as a disciplinary research topic has been resisted almost reflexively, even while the complexity of econometric ideas has increased over the years. Economists seem to feel instinctively that the specific algorithmic aspects of econometric software most properly should be considered from the perspectives of computer science or operations research, leaving to the economic specialist, the econometrician, only the task of specifying the particular formulas to be implemented algorithmically, but not the assessment or evaluation of the end result nor the manner of its implementation. What may be most indicative of the tendency of economists to regard econometric computing as logically separable is that both textbooks and the more general econometric literature typically only refer to the existence of software, without considering its particular aspects and nature. The specific issues of software creation, including even those most obviously 'econometric' in content, have seldom been considered and evaluated publicly by economists. It is therefore not surprising that most economics and econometrics journals exclude discussion of these matters. Indeed, economic journals normally reject as not germane articles that focus too obviously on specific computational issues, even when critical to the evaluation of the published findings. A possible exception to this is numerical accuracy [175, 253], but only recently and still in token form [34, 38, 154, 167, 169, 170, 172, 174, 219, 229, 231, 232]. To date, economists deal with computational issues at a full arm's length and depend upon others, often statisticians and the statistics literature, to probe the details [10, 122, 155, 220, 239], even if some related issues have been considered within the bounds of econometrics [16, 17, 21, 22]. Most economists are aware of John von Neumann's role in the development of game theory, but comparatively few know that he also wrote the first computer program and participated in the design of the first stored-program computer [140]. Of these several endeavors, which in the end will prove the most seminal is an interesting question.

1.2 The nature of econometric software

The phrase 'econometric software' is actually a comparatively new term in the economics literature. Its newness can be assessed using JSTOR, the full-text-searchable online journals archive. A query of the 52 economics and econometrics journals identified there as 'economics journals' reveals that, since 1950, only 182 articles have been published that include the term 'econometric software.' The first use of this term in print appears to have been by Robert Shiller in a 1973 *Econometrica* article [217]. It appears in only 14

articles published prior to 1987, fewer than 25 before 1990, and 94 during the 10 years since 1995, the latest time for which JSTOR results are available at the time of writing. Evidently, this phrase has only recently begun to emerge into full disciplinary consciousness. It is nonetheless more popular among economists than the seemingly more general phrase 'statistical software,' which occurs in only 85 articles published since 1950, the first appearing in 1982. Of course, the JSTOR journals represent only a proportion of those classified by the *Journal of Economic Literature* as economics journals. Therefore, these findings are hardly conclusive, but they are suggestive.

The currency of this term is merely interesting. Of more substance is what it means. To determine meaning, several preliminary matters need to be considered, the most important being the meaning of 'econometrics,' which is also not obvious. Judging on the basis of the chapters of this handbook, 'econometrics' should be interpreted more broadly here than is typically the case in any modern econometrics textbook or even in 'econometric' journals. Textbooks treat 'econometrics' essentially as the subdiscipline of economics concerned with parameter estimation and closely related topics. Moreover, they treat it somewhat abstractly, giving wide berth, as indicated, to any attempt to translate 'econometric ideas into software.' Furthermore, neither they nor mainline econometrics journals include the solution of nonlinear systems of equations as a standard topic, even when characterized as 'econometric models.' Nor do they examine a number of the other topics that are addressed in this volume.

Nevertheless, the meaning and scope of 'econometrics' has been visited regularly since 1933, when, in the context of an editorial introducing the first issue of *Econometrica*, Ragnar Frisch proposed a rather broad definition that emphasized the unification of economic theory, mathematics, and statistics. Clive Granger, in a recent comparative review of various textbooks [93], has since suggested that the Frisch proposal 'now sounds like a definition of research economics' (p. 116). He argues instead for the definition advanced by Andrew Harvey [105] that 'econometrics is concerned with the estimation of relationships suggested by economic theory,' which, Granger indicates, was also rephrased by Greene in the 1994 edition of his textbook. Granger goes on to suggest (p. 117) that a 'more pragmatic viewpoint is to say that econometrics is merely statistics applied to economic data,' which leads him to ask the question: Why, then, do economists not just use statistics textbooks? His answer is that 'we need a special field called econometrics, and textbooks about it, because it is generally accepted that economic data possess certain properties that are not considered in standard statistics texts or are not sufficiently emphasized there for economists.' However, he does not continue this line of thought, which, if pursued, would lead to the rather interesting and probing – if possibly vexatious – question whether, collectively, econometrics textbooks simply reflect the current content of econometrics, or if, instead, through selection and emphasis, they inevitably channel and shape its progress? Resisting the temptation to grapple with such matters, Granger limits himself to a review of the contents of that particular textbook generation.

In line with the spirit of Granger's ruminations, one way to specify 'econometric software' is to say that it is what economists do with economic data that defines this software – that is, it is computer code that happens to be applied to economic data. But, what economists do with economic data presumably shapes the discipline of economics over time and, in the process, questions might be raised that can cause econometrics to evolve in order to answer them. One way to consider the actual progress of econometrics

since 1933 is to observe it through the prism of Frisch's other famous editorial dictum [84, p. 2], that:

> Theory, in formulating its abstract quantitative notions, must be inspired to a larger extent by the technique of observation. And fresh statistical and other factual studies must be the healthy element of disturbance that constantly threatens and disquiets the theorist and prevents him from coming to rest on some inherited, obsolete set of assumptions ... This mutual penetration of quantitative economic theory and statistical observation is the essence of econometrics.

Fresh factual study of the way in which econometrics is practiced is a way of providing a healthy element of disturbance.

Possibly the most direct way to determine the nature of 'econometric software' is to recognize that quantitative economic research establishes operational requirements; in particular, it involves at least four conceptually distinct activities:

- data acquisition;
- data organization and initial analysis;
- selection of particular computational tools, possibly including choosing one or more analytic software packages or even the creation of specific software; and
- using these tools to perform this research, that is, to generate and display the results.

Of course, with the advent of the Internet, especially during the past 10–15 years, all these activities now demonstrably relate to the development of software even if they are not necessarily undertaken in strict sequence nor always pursued as separate and distinct tasks. Sometimes, some of them may be minimized, even to omission. But it is also possible for most of the analyst's time still to be spent on the first three activities, as was common before the advent of the microcomputer – or indeed even before mainframe computer software became widely available.

Once these activities begin to be considered in detail, certain definitional gray areas inevitably start to emerge, such as the common use of commercially available software not econometric by design in order to perform some research-related operations – including word processing, spreadsheet, and presentation software packages, not to mention analytical packages imported from other disciplines. At some stage, this matter of ancillary software might need to be probed. After all, Excel can perform regressions [23, 176–178] and Matlab can solve econometric models [116]. Economists, if not always econometricians, employ both programs, neither of which was specifically intended for their use. However, it is initially much simpler to ignore software not explicitly created by, for, or at the behest of economists or econometricians, even if this software may become quasi-'econometric' when employed by econometricians or applied to econometric tasks. The justification for turning a blind eye can be that it is software specifically designed as 'econometric' that is likely to express distinctive disciplinary characteristics, although, if economists characteristically tend to use 'other' software, one of the implications might be that econometrics could be in the process of evolving in another direction.

It is thus tempting to define software by the purpose for which it is created, but this too can lead to difficulties. As indicated, in the beginning, the electronic computer was often seen as simply a faster means to make specific calculations [36]. The importance of its speed, compared to electromechanical devices or hand calculation, was immediately recognized, both by governments and by other funding sources, as well as by those who created the machines – including von Neumann, whose original interest in computers might have been a result of his association with the Manhattan Project. A direct comparison, in the 1940s, of the speed of the electromechanical Mark I at Harvard with the electronic ENIAC at the University of Pennsylvania (the first, but not stored-program, electronic computer) had revealed that the ENIAC could perform individual calculations up to 900 times faster [2, 41, 248]. The ENIAC was later used to make calculations related to the development of the hydrogen bomb.

This initial focus on individual calculations sometimes had a particular effect on machine design. In an historical hardware review, Rosen [214, p. 14] points out that 'the early scientific computers [as opposed to data processing equipment] were designed in accordance with a philosophy that assumed that scientific computing was characterized by little or no input or output.' However, even in this context, it was soon discovered that the number of calculations made tends to increase, finally forcing a more balanced machine configuration. Generality of use, and broader capabilities, rather than specialty of purpose, came to be seen as desirable, perhaps because of a perception (not always correct) of declining marginal cost. Consider, in this context, the Wang and other, now-almost-forgotten, dedicated word processors of the late 1970s, which have since given way to the much more generally defined personal computer.

How much generality mattered in the beginning is, however, questionable. The use of the electronic computer by economists appears to have been driven initially by a desire simply to make particular calculations. For example, Robert Summers and Kazuo Sato in the very early 1960s [197, 235], Arnold Zellner and Art Stroud in 1962 [256, 257, 259], Harry Eisenpress in the early 1960s [61, 62], James Durbin and Mike Wickens in 1964 [59, 247], Denis Sargan in 1964 [110], and a group of A.R. Bergstrom's graduate students over a number of years in the 1960s [194], as well as Ross Preston [197] during much the same time, quite independently all wrote, or were closely associated with the writing of, programs to implement or study specific econometric techniques – but apparently not originally for the conscious purpose of writing 'econometric software' for it's own sake. In contrast, others, among them Robert Hall, Michael McCracken, and Clifford Wymer in the mid-1960s [25, 26, 160, 162, 163, 165, 192, 204, 211], and David Hendry at the end of the 1960s [114, 115], wrote programs the direct descendants of which still exist and which affected the development of other programs. Demonstrably, each of these economists, as well as others, consciously sought to develop more broadly defined and generally applicable software that has evolved over time, regardless of whether this destiny was at first foreseeable. From the perspective of a fly on the wall of their keypunch rooms (where programs were encoded in the form of decks of cards or paper tapes with holes punched in them), it might not have been obvious at first who would continue beyond the first working program and who would not. However, those who persevered made the transition, much as a pioneer who, during his first week in the wilderness, builds a lean-to that, with adherence and time, ultimately becomes a house and acquires features that embody and display finished design elements.

1.2.1 The characteristics of early econometric software

Of course, the point in this process at which the design of the lean-to or its successors becomes architecture is a matter of judgment. Similarly, during the primordial stage, there is no essential difference between programming a computer and developing software. Whatever the motivation, creating computer programs that perform useful econometric calculations requires a considerable amount of work. Initially, it also involves a large dose of learning by doing. For the novice, this first stage usually represents an attempt to achieve a specific task and, as indicated, in the 1960s it was common for the early stages of a software project to involve first the construction of a series of separate, essentially single purpose programs to be used in concert. For instance, at the Brookings Institution in the context of the US Quarterly Model Project [57], as late as 1968 the process of performing a regression involved executing sequentially three separate programs: the first to form or update the databank, the second to perform the entire set of pre-regression data transformations, and the third to compute and display parameter estimates. However, about that time, these functions began to be integrated. The then relatively widely used ECON program provides an example. It was created by Morris Norman, on the basis of earlier work by Ross Preston [197], originally to support the estimation of the Wharton models and related work. By design, ECON used portions of the same code to perform ordinary least squares, two-stage least squares, and limited information maximum likelihood. This program also incorporated data transformation subroutines, thus in effect absorbing the work of the first two of the three Brookings Model Project programs described earlier.

However, this account possibly conveys too great a sense of linear development, for technologies often evolve in a somewhat helter-skelter fashion. At this time, as indicated, computer programs almost always came in the form of punched cards: 'jobs' were typically submitted in the form of decks of cards, or in some cases paper tape, that included the program source code and the sample data. Some 10–12 years previously, computers, such as the IBM 650 [7, 237], both read cards and produced their output in the form of cards. By the mid-1960s, output was instead more commonly produced in printed form, on large, accordion-folded, connected sheets of paper, but cards or paper tape were often still used for data input. In 1968, in contrast, the separate programs being used for the Brookings Model Project, written or modified by John Angstrom, Jim Craig, Mark Eisner, and possibly others, both wrote and read magnetic tapes, reflecting the substantial data requirements of a large macroeconometric model, the largest of its time. Since these data were machine resident, the 'jobs' at Brookings usually consisted only of source and specific-operation code decks preceded by job control cards, rather than hundreds of data cards in a deck. Thus, the programs of that day frequently might exhibit quite rustic features, combined with others that at least anticipated a later state of the art. However, generalizations about such software can be overdone, as there was at this time only a bare handful of programs developed to support applied economic research by groups of economists.

The programs of the period before 1969 seldom included obvious 'architectural' design elements in their construction. The TSP program is an exception, for, starting in 1966–67 as a result of work by Mark Eisner and Robert Hall, its design involved the modular organization of subroutines to perform specific tasks, in contrast to an earlier

tendency to write a single main routine, with perhaps one or two task-specific subroutines [100, 102]. Other economists, including Charles Bischoff, Ray Fair, Robert Gordon, and Richard Sutch, each wrote specific TSP subroutines or incorporated particular econometric techniques. These pre-1969 versions of TSP also included the first vestiges of a symbolic language interface [203]. With a few exceptions – such as MODLER [209] and TROLL [63] – until the later 1970s the operation of other programs of all types was commonly controlled by numbers, letters, and labels located in fixed fields on punched cards or paper tapes. It took considerable time then simply to operate a program, given the need not only to program, but also to validate results on a case-by-case basis.

A further difference is that in those days software used by economists was neither leased nor sold. It was shared: transmitted in the form of source code, and then compiled for individual use, sometimes at the beginning of each run. Source code for some of the statistical regression packages of the 1960s, such as for BMD and OMNITAB, still exist in a few copies; therefore it remains possible to make comparisons. For the most part, their individual characteristics are not strongly distinctive. Whether a particular program of that period might be judged to be 'statistical' or 'econometric' is rather subtle, a matter of the specific techniques – most often basic regression – programmed into it. A program of that period that offered ordinary least squares, weighted least squares, and stepwise regression is, on the strength of this information alone, hard to classify definitively by discipline, but one that included two-stage least squares was likely to be econometric, rather than something else.

Clearly, what began to distinguish econometric software was the progressive incorporation of features that were designed primarily to support the interests of econometricians, rather than applied economists generally. Recall, however, that in the 1960s econometrics was not as distinctly a separate subdiscipline of economics. In that day, only Malinvaud's textbook [158] provided the student with a serious econometric challenge. Textbooks by Johnston in 1963 [128], Goldberger in 1964 [88], and Christ in 1966 [42] were formidable only to the degree that the reader lacked a reasonable knowledge of linear algebra – and, actually, in the interest of making econometrics more approachable, Christ's did not require even this. In any case, the econometric knowledge needed to understand, or even to write, any econometric software package of the 1960s would fill no more than 150–200 textbook pages, if that. The software content was generally limited to a few standard estimation techniques and a handful of supplementary statistics, such as t statistics, one or two types of R^2, standard errors, the Durbin–Watson, and the like. It was not until the 1990s that more than a small proportion of the then known misspecification tests were commonly implemented [204, 210].

A more significant barrier to software development in the 1960s was programming language facility, usually writing Fortran – not simply the ability to write a few lines of code, but rather knowing how to organize the coding of a regression program procedure, including matrix multiplication, matrix inversion, how to print a portion of a matrix on a line printer page of output, and other such programming techniques. This need to program was to a degree alleviated by the creation of subroutine libraries by IBM and others, but the lack of books on the subject made general programming more difficult than might be immediately obvious in retrospect. The shared source code listings, although easily obtained, often lacked embedded comments and could be opaquely written. 'Use the source, Luke.' Although, of course, a phrase current only since the 1980s was in the 1960s often the only option. This era was also an important development period

for programming techniques, involving almost always an element of original design, as indicated earlier: for instance, such as whether to use an entire dataset to compute the sums of squares and cross-products matrix, then selecting the relevant submatrix to produce given parameter estimates, or instead to form this submatrix directly. Computers in those years were comparatively slow, so that such considerations could be important.

1.2.2 The expansive development of econometric software

It is important to recognize that, at the beginning of the 1970s, there was an evolutionary fork in the econometric software development road. On the one hand, at the end of the 1960s, the first econometric modeling languages [203] began to be written, in order to support the creation, maintenance, and use of large macroeconometric models. The creation of multi-hundred equation models, such as the Brookings and Wharton models in the USA, the Candide model in Canada, the Cambridge growth model in the UK, and others elsewhere [31], implied the need to manage time-series databases containing thousands of variables, to create model solution algorithms [197, 216], and to develop graphical, tabular, and other easily understood display facilities. An economic consulting and forecasting industry, including such firms as Chase Econometric Associates, Data Resources, and Wharton Econometric Forecasting Associates, and an emerging time-sharing telecommunications technology, both fostered and financed the creation of quasi-natural-language software packages offering substantial database maintenance facilities, such as Damsel, EPS, MODLER, and XSIM [206]. These packages' specific econometric flavor, although ostensibly their calling card, was in practice often less important to their users in government agencies, firms, and other organizations than other features that made them, and the data they provided, generally accessible: many economists were neither skilled as computer programmers nor had previously used the computer. The result was to mandate an interface as close as possible to a stylized natural algebraic–mathematical language, such as might be typed on a sheet of paper. Because of the typical user, time plots and attractive time-oriented tables, as well as the ability to make data transformations, were primary desiderata, much more so than the ability to perform complex nonlinear parameter estimation.

On the other hand, particularly at MIT, as documented in the pages of the short-lived NBER journal *Annals of Economic and Social Measurement*, there was a concerted and coordinated effort to develop TROLL, including Gremlin [19, 63, 64], which contained linear and nonlinear regression algorithms. This work also affected and stimulated the ongoing development of TSP [29, 98, 99, 101]. The development of TROLL as a generalized package was originally driven by the aim of producing an econometric modeling language, with a programming language-like interface to support the user development of macros. Underlying this work was research on generalized and efficient numeric procedures [19, 21, 22]. The development of these programs at the MIT Center for Computational Research in Economics and Management Science [1] fostered the implementation of a variety of parameter estimation facilities [29, 37, 71, 106, 107, 147] and, to a degree, also fed back into the economic consulting industry, inasmuch as XSIM was initially developed as an extension of TROLL. At the Center, under the direction of Edwin Kuh, economists devoted considerable effort to the study of relevant computer algorithms [29, 48–50, 120, 147–150] and closely related regression diagnostics [19, 21, 22]. It was following such advances that increasingly

sophisticated, wide-area online telecommunications-linked economic database, analysis, and econometric modeling systems were developed during the 1970s, which by the end of that decade became used worldwide [3, 4, 56, 205, 206].

Elsewhere in the 1970s, such as at the DAE in Cambridge and at the London School of Economics (LSE), econometric software was being purposefully developed in academic environments either to support the creation and use of large, multi-sector models – specifically in the form of the Cambridge growth model [14, 189–191] – or else to enable the estimation of dynamic econometric specifications and, more generally, improve the methodology of specification search [186–188]. At the LSE particularly, work on specification search led ultimately to the development of GIVE and, later, PcGive, as software designed to foster the general to specific, or LSE, method of specification selection [110, 112, 113, 115, 146, 181]. In this context, the development of software played an integral part in the intended improvement of econometric methodology, with this software specifically designed as an analysis tool to be used pedagogically. At other places in the UK, with the establishment in the 1970s of the London Business School Centre for Economic Forecasting, and econometric modeling projects at Liverpool, Southampton, and other universities and research organizations, such as the National Institute of Economic and Social Research and the Treasury [13, 238], other software packages were developed, among them CEF, NIMODEL, and REG-X, as well as software developed under the aegis of the ESRC Macroeconomic Modelling Bureau at the University of Warwick [240–243].

In other parts of the world, for instance in Canada, the development of software such as MASSAGER and SYMSIS by McCracken and others [160–165] to support the construction and use of the Candide and, later, TIM models [30] increasingly took the form of econometric software designed to support the use of large macroeconometric models. More generally, particularly with the establishment of the LINK Project [47, 117, 137, 183, 198], which fostered the coordinated use of macroeconometric models of countries worldwide, more model-related software was developed, although for the most part poorly documented and not always representing independent software development efforts. In addition, other software creation by individual economists also occurred, but this quiet, almost secretive, initiative, also under-reported in the mainstream economics and econometrics literature, is now difficult to evaluate, except in those cases in which programs created then later became widely used. From the reports of the use of microsimulation models and the growing quantity of economic datasets, often micro-datasets, as well as numerous published articles that provide empirical results related to a broadening range of economic and financial theoretical topics, it is obvious that software was both frequently employed and sometimes created in the research process [9, 24, 65, 108, 215]. There is also evidence that SAS, SPSS, and other statistical software packages were commonly used by economists, rather than specific econometric software [204].

The creation of software by individual economists in the 1970s, particularly for personal use, can be viewed as taking two forms. In a few cases, individual economists, such as Ray Fair [70, 72–76], Ross Preston [197], and Clifford Wymer [25, 27, 28], developed software related to model building and use. Others, at least in the first instance, focused upon the development of software related to parameter estimation, among them Bill Greene, Bronwyn Hall, Hashem Pesaran, Peter Phillips, David Reilly, Chris Sims, Lucy Slater, Houston Stokes, Kenneth White, and Arnold Zellner [95, 96, 188, 194, 204, 211, 230, 233, 234, 244–246, 254, 255, 259]. The resulting software nevertheless displays

considerable variety. Some developers focused on microeconometric applications and hence the use of cross-section and panel datasets, as well as specific, associated estimation techniques. Others focused on particular methodologies, such as Bayesian estimation [196] or Box–Jenkins and other time-series techniques. Certain of these efforts resulted in software packages that still exist and continue to be developed today. A recently published compendium [211] describes these programs individually and cites pertinent manuals and other documents that still exist. An associated comparative assessment of the architectural and other characteristics of these packages, based upon an interactive survey of econometric software developers, provides further information [85, 204]. Additional software development of course occurred during these years in the context of other economists' personal research, often resulting in published articles, but this software generally went unremarked and is now difficult to assess properly. In contrast, there has been more of a tendency during the past 10 or so years to consider explicitly the structure of algorithms, although these contributions sometimes appear in specialist journals not likely to be read regularly by econometricians [80–82, 86, 141–145].

As well as the work just considered, some software development that began in the 1970s involved the development of specific types of model solution software, including that to support control theory applications. These applications were often not well documented, especially when the associated software was not made generally available, but interest in the topic is evident in the papers published in two special issues of the *Annals of Economic and Social Measurement*, on control theory (1972, vol. 1, no. 4) and on control theory applications (1977–78, vol. 6, no. 5). Subsequently, specific software developments have been documented in the broader literature [45, 53–56, 75, 76, 121]. Since the late 1960s, nonlinear econometric models, a category that includes virtually all macroeconometric models, have been solved using Gauss–Seidel, Jacobi, and Newton techniques. In the first case, to solve a model usually requires it to be ordered at least quasi-recursively, a somewhat heuristic process considered typically in the specialist literature [32, 79, 125, 126, 163, 184]. When using Newton techniques, the model need not be reordered but may have to be linearized, at least in part, in order to obtain the partial derivatives needed to compute the Jacobian used in the model solution, although later versions of TROLL and TSP provide nonlinear differentiation procedures that to a degree obviate this need [55, 216]. Optimized solutions, particularly in the context of rational expectations or model-consistent solutions, add a further level of complexity to the solution process [75, 76, 78]. All these solution techniques are more difficult to implement accurately than is generally recognized, with a number of potential pitfalls [173]; however, the numerical accuracy of model solution software has seldom been tested [203].

Given this consideration of model solution techniques, the implied definition of econometric software must be expanded beyond what can be justified by specific appeal to the mainstream econometrics literature. With the notable exception of Klein's, econometric textbooks usually have not ventured into these potentially dragon-filled seas, and it is generally to journals such as the *Journal of Economic Dynamics and Control* and *Computational Economics*, rather than the *Journal of Econometrics*, say, that one must look to find any discussion of these techniques, particularly during the past 15–20 years. During the 1970s, *Econometrica* would occasionally publish articles on this subject, but not recently. Therefore, once having embarked upon considering the issue of model solution techniques, we have left safe harbor and, by implication, have adopted a definition of

econometrics that includes not only the estimation of econometric specifications, and associated tests, but also the use of models. It is natural, when writing an article, or even an econometrics textbook, to shape the content to fit what is best defined and understood. In contrast, software development exerts a pull into the unknown and uncharted, forcing decisions to be made in order to satisfy the demand of completeness.

It is thus important to recognize that, by and large, the coverage by econometrics textbooks of computational issues remains that of an earlier age in which economists simply ran regressions, albeit sometimes involving simultaneous equation estimators. Furthermore, even the most modern textbooks present their material almost as if the reader will be involved personally both in making the calculations and in their evaluation so as to assess a range of candidate specifications. The essential logic of this pedagogic approach is easily appreciated: the student econometrician needs to understand the characteristics of the various parameter estimators and misspecification statistics, as well as the contextual implications and possible pathological conditions. However, the reality is that the typical economist does not calculate number by number, but instead uses whatever software is readily available. A drawback of this is that the economist is then generally unable to determine exactly how the computations are made. Of course, as recently demonstrated [174], self-calculation never necessarily implied an absence of problems, but it did provide the economist close contact with the computational process and forced a degree of step-by-step training. What is lacking today, as a necessary substitute, is guidance concerning the relationship between the theoretical constructs and the computational realities. Once having tasted of the fruit of the tree of econometric knowledge, the reader is in effect rudely cast out of the Garden of Eden, to make sense alone of the world as it is. Not only is there seldom an attempt made in textbooks to explain the particular implications of the intermediation effect of the use of software, but textbooks also continue to convey that the choice of software can be made arbitrarily, without concern for any qualitative issues or of a given program's suitability for a particular application. In fact, once a software package is chosen, even in the best case the user is almost completely dependent upon the specific provisions made by its developer. If only simple linear techniques are considered, the difference between the textbook world and the less-than-ideal real world may not always be significant, but in the case of nonlinear estimation, in particular, the gap can be wide indeed [172, 173, 219].

That such gaps can exist is an important underlying issue, for it still seems to be widely believed that the existing software necessarily meets the appropriate qualitative standards. Reflecting this belief, when economists evaluate software, they often appear to adopt the point of view that software should be judged simply by its ability to facilitate use on the user's terms, defined by the particular tastes and even prejudices of that user. McCullough and Vinod [175], in particular, have argued (p. 633) that economists 'generally choose their software by its user friendliness,' going on to suggest that 'they rarely worry whether the answer provided by the software is correct.' This suggestion's most thought-provoking aspect is not only the economist's possible presumption of numerical accuracy, without evidence, but also its assertion that, for at least certain econometric software users, the perceived currency of the interface is the deciding factor, in part perhaps because of an ignorance that has been enforced by the silence of the mainstream literature. An equally disturbing aspect may be the level of user expectations it implies, in a world in which economists typically wish to obtain software cheaply. Inexpensive software is of course generally associated with increasing returns to scale, but

it is also possible that the economics profession might be too small a market to support the cost of providing all-singing, all-dancing, custom-built econometric software, comprehensive and competently produced, accurate to a fault, and yet so well designed with an always up-to-date interface that it can be competently evaluated on the surrogate basis of superficial characteristics alone.

However, to provide the proper perspective for a more detailed evaluation of the implications of these circumstances, it is first necessary to place them in their appropriate historical context. The mention of microcomputers here is potentially misleading, given their ubiquity today, for although it is true that the first microcomputers became available to 'hobbyists' as early as 1975, the 1970s were, as a general proposition, still very much the age of the mainframe, for economists as well as most other people. Only at the end of this decade did economists begin to write software for the microcomputer [204]. Furthermore, many of the earlier econometric software initiatives mentioned took fully a decade to come to term, so that considering 1970s computer use in context it is important to recognize that this decade's predominant computational characteristic was not only a mainframe focus but also a continuing relative rarity of use. The 1960s were a time during which economists more generally became computer users, as suggested earlier, but this was a change relative to the use in the 1950s. Even in 1970, only a small number of economists, or people at large, had begun to directly use the computer intensively. In the case of economists, many were then graduate students, and of these only a proportion customarily spent each night and weekend in the keypunch room. Today, in contrast, computer use of course begins at kindergarten, or even before, and extends to any age pervasively, but such behavior began, at the very earliest, in the late 1980s.

Today there is also stress upon user accessibility, which did not exist then to anything like the same degree. The human interface is now much more developed, in a way that permits the user to operate programs at a much higher level of abstraction. In contrast, as indicated above, programs in the early 1970s often, and in the later 1970s sometimes, still needed to be operated by placing numbers in fixed fields of punched cards, a given integer number from 1 to k indicating which of a particular k options the user wished to select, or by the use of 0 to indicate omission. Sometimes data transformations needed to be made explicitly, the user coding these in Fortran or other high-level language. However, as discussed, in the 1970s there were also certain incentives that led some individual econometric software developers to begin to focus on the human interface and on abstract symbolic processing, as well as large-scale data management, whereas others considered particular algorithms or programs that have effectively developed as representative of what might almost be considered econometric 'schools' of thought, even if their advocates might have sometimes gathered in one-room schoolhouses. Most commonly, in those days, economists and econometricians created or modified particular programs for their personal use, only certain of which ultimately became widely used.

Particular examples of programs of the first type, the interface focused, include Damsel, EPS, MODLER, and XSIM, only one of which is still maintained but each of which were then associated with the creation, maintenance, and use of relatively large macroeconometric models. As touched upon earlier, the use of such models characteristically involves the need to create and maintain substantial time-series databases. It also requires the processing of symbolic data in the form of equations, and imposes a requirement for software that can be used even by teams of people. Examples of programs of the second type, which can be regarded as interesting because of their particular internal

algorithmic characteristics in the 1970s, include B34S, TSP, and WYSEA. Programs of the third type, those that can be construed as individually associated with a particular econometric 'school,' include such diverse packages as AUTOBOX and AUTOREG (represented today by its direct descendant PcGive). More broadly, programs created originally in the 1970s for local, even personal, use, but that have since been developed for use by economists generally, include those such as AREMOS, CEF, FP, IDIOM, LIMDEP, MicroFit, ModelEasy+, RATS, REG-X, SHAZAM, Soritec, and WinSolve. Some of these might also be regarded as being assignable to a 'school.' These classifications should all be regarded as being only tentative, but they nevertheless illustrate aspects of econometric software development pre-1980.

During the 1970s, there were also larger forces propelling the development of econometric software. Overall, the economic spirit of that time was distinctly activist. At the end of the 1960s, in keeping with the particular prevailing Keynesian paradigm, not only was there some feeling among economists that the economy was potentially precisely manageable (although possibly not to the degree that it has been represented since), but – just as importantly – there was also a broader willingness on the part of government officials and corporate leaders, particularly in the USA, to believe in the capability of the economist to 'fine-tune' the economy. All this was, to a degree, a consequence of the fact that then the memory of both the 1930s depression and the escape from it in the 1940s and 1950s was still vivid. In 1945, it was popularly believed, a belief shared by many economists, that an economic downturn was likely, that World War II possibly represented merely a temporary period of 'full employment' that would inevitably give way to widespread unemployment when 'the troops came home' [135, 185, 251]. However, 25 years later, at least in the case of the major industrialized countries, each of the recessions that had occurred had proved to be of short duration and at worst represented small fluctuations about a distinctly upward trend. One of the consequences, in 1970, was substantial forthcoming corporate and government support for the creation of economic consulting firms, such as Data Resources, Chase, and Wharton, each of which began to create and support large-scale, computer-resident economic databases and econometric software systems [203, 206].

The development of econometric software during the period between 1960 and 1980, as it has been traced here, can in fact be seen as reflecting two distinct 'motivating forces' – those internal to econometrics, and those external, deriving from economics as the parent discipline. In the case of individual econometricians, this generally took the form of software development that was focused upon parameter estimation using a variety of estimators; the academic imperative driving this innovation was of course the usual desire to present the new and different. In the case of the major economic consulting and forecasting firms, the imperative was to provide billable services to clients; that is, to applied economists in governments, corporations, and other organizations. Software development in this context and at that time was usually competitively motivated by the goal to provide time-sharing software services, in combination with access to substantial time-series economic databases, in many cases via telecommunications links [5, 180, 206]. As time went on, an important aspect of the software development supported by economic consulting firms was the creation of a semi-natural-language, symbolic human command interface of the type prevalent even today, intended to be easy to learn and easy to use [4, 56, 130, 179, 203]. Another important aspect, reflecting the widening range of users, many of whom might be less likely to call themselves 'econometricians'

than simply 'economists' or even 'planners' or other occupational descriptions, was the development of more broadly based facilities to support onscreen tables, graphs, and even maps, although not at 21st-century standards.

Both these types of software development efforts have proved to be important ultimately, each in its own way. Individual software development, for self-consumption, does not normally result in programs that can be used easily by other people, no matter how econometrically interesting the algorithms created. In contrast, programs developed for a broad user group, of necessity, tend to offer not only 'user-friendly' interfaces, but also data display capabilities that can actually be quite important in applied research. The academic payoff from software development, such as it is, tends to be much greater for work that leads to the publication of information about new econometric technologies or stimulates the development of new economic theories, but this represents a private, not necessarily a social, benefit, beyond the creation of new knowledge that may or may not be of substantial or wide interest. In contrast, the greatest social benefit may arise from new computational technologies that benefit the research of all economists and econometricians alike. However, it is also important to recognize that, for applied economic research, and the econometric software development that supported it, the 1970s were years of substantial aspiration, yet also a time during which the economist's reach exceeded his or her grasp. One of the reasons why this aspect needs to be considered is that economists, in their introductions to journal articles and other publications, often provide stylized, frequently not particularly well founded, historical statements, intended as motivators for the argument to be presented, and not always representing deep belief, even the author's. Yet these statements, as historical assessments, can be quite misleading, and can then also be cited and passed on as gospel, finally entering into the folk wisdom of the discipline, essentially in the form of old wives' tales.

1.2.3 Econometric computing and the microcomputer

Among those used by economists, the machines prevalent prior to the early 1980s ordinarily can be classified as mainframes or minicomputers, although supercomputers had by then also appeared in increasing numbers, after being introduced in the 1960s, even if seldom used by economists. Mainframes were of course intended to be used 'enterprise-wide,' whereas minicomputers were meant to be used in departments and other, generally small, subclassifications of organizations. Supercomputers, in the 1970s typically a category of fast, vector processor machines, commonly had the interesting characteristic that mainframes were generally used as auxiliary processors for the input of data and output of results. However, irrespective of classification, the most fundamental characteristic of computing before the early 1980s was not only that the computers used were almost without exception organizationally owned, rather than personally, but also that they were shared. Mainframe sharing at this time implied that, however fast the machine might be when operated by a single person, it could be quite slow for the average user. In particular, it might be slow because it operated in batch mode, with programs prioritized and queued as they were read in, and sometimes not executed for hours. On output, the hard-copy results were sorted and distributed by hand by the computer's one or more operators. The effect could be turnaround times of an hour or more, or even 24 hours for 'large' jobs, those requiring memory in excess of 512 kB, or sometimes as little as 256 kB. Even when machines were used in 'time-sharing' mode, the fact that

individual users' 'jobs' were almost always prioritized and then queued, and also might require a human operator to mount tapes and removable hard-disk units, could mean that even minutes passed between the entry of a single command and the computer's response. The first personal computers were themselves slow machines, compared to either mainframes or minicomputers (or personal computers today), but they were single user and self-operated, and in practice these characteristics caused them, even as early as 1981, to be time competitive with mainframes and sometimes even supercomputers.

The fundamental problem that the microcomputer initially posed for the econometrician was the lack of useful software, which almost always occurs when hardware characteristics are fundamentally changed because of the adoption of an entirely new central processing unit (CPU). In addition, the architecture of this new class of computer at first also represented a significant step back in capabilities: maximum memory size on the order of 64 kB, rising to 640 kB only more than a year later, and small, slow diskette drives for permanent storage rather than hard disks, with hard disks initially unavailable and then reaching the size of 20 MB, as a common characteristic, only in 1984 – not to mention CPU operating speeds of 6–8 MHz or less before 1986 [39, 40]. Furthermore, it took several years before microcomputer software provided the capabilities of mainframe software, reflecting that writing software takes time, but also that the language compilers and linkers available for the microcomputer were at first 'bug' ridden, idiosyncratic, and originally designed for small, memory-undemanding programs. In at least one case, in 1982, it was necessary to 'patch' an existing linker in order to make it possible to convert an econometric software package from the mainframe to the PC.

Nevertheless, by the fall of 1983, at least one econometric software package capable of estimating and solving (small) econometric models was available for the IBM Personal Computer and compatibles. In September 1984, the first microcomputer-based economic forecasting service was introduced at the annual meeting of the National Association of Business Economists, combining a 250+ equation Wharton Quarterly Econometric Model of the United States with the MODLER software [209]. The solutions for a 12-quarter forecast horizon took less than 4 minutes. This software had the same capabilities as its mainframe version [56]; in particular, it could then be used to create, maintain, and solve econometric models of as many as 1000 equations. By the end of 1985, the other packages available for the 'PC' included AREMOS, AUTOBOX, PcGive, RATS, Shazam, Soritec, and Stata, as well as limited versions of both SAS and SPSS. Even earlier, a few, relatively limited packages had been implemented on both a Tandy machine and the Apple II [204], including interestingly enough a program called 'Tiny TROLL,' created by Mitch Kapor at MIT, parts of which were then incorporated into the VisiCalc spreadsheet package and subsequently also influenced aspects of the development of Lotus 1-2-3, and later other packages, such as Excel.

In order to chart the subsequent evolution of econometric software during the present microcomputer age, it is possible to proceed by tracing the broad outlines of the computational developments of the past 20–30 years. The computational shift during the 1980s, from the creation of software and systems on large institutionally based machines to the use of the personal computer as the locus of such work, can be viewed as responsible for the range of econometric software that exists today. The personal computer, because of its affordability, wide distribution, and steadily increasing capabilities, not only provided an important developmental context but also became the basis of an extensive market for this software. The 1960s may have been the time that economists generally first began

to learn to use the computer, but pervasively it was in the later 1980s that computer use began to become truly widespread. The comparative degree of market extensivity is even more obviously apparent today, given the ubiquity of the notebook, or laptop, computer, and otherwise the sheer number of personal computers commonly found in offices and homes, not to mention such things as the recently accelerating convergence of television and computer technologies. Of course, in addition, the development of the Internet as an effective successor to the more local, mainframe-based wide-area networks of the 1970s has obviously also had a significant impact, particularly from the second half of the 1990s, especially on the distribution of economic data and information.

Consequently, although it is possible to talk in terms of nearly 60 years of evolution, in truth the impetus for the development of today's number and variety of econometric software packages is more recent. Their present characteristics are essentially the direct result of a combination of relatively modern circumstances, among them being the introduction of the microcomputer in the 1970s and 1980s, the simultaneous expansive development of econometric techniques since the 1960s, and most recently the increasingly common adoption of a graphical interface, often while preserving macro language capabilities, in conjunction with the more and more widespread use of the Internet since the 1990s. Among the effects, the broadening and deepening of econometrics – and, more generally, quantitative economics – especially during the past 30 years, has had a significant impact on the range of the present-day properties of these programs, resulting in considerable diversity. For example, functionally classified, today they can be placed in categories that include basic regression, advanced estimation, and econometric modeling languages. Considered in terms of characteristic functionality and interface, they can be classified as ranging from those defined by specific-selection, menu-oriented econometric features to algebraic quasi-natural-language econometric modeling and programming languages that provide even the capacity for an individual user to create new techniques [204].

Simultaneously, substantive changes in hardware have of course occurred during the past 20 years. As indicated, the personal computer in 1987 operated at 6, 8 or 10 MHz; today, many modern notebooks operate at or near 2 GHz or better. The 1985 computer ordinarily contained at most 640 kB of easily accessible memory; the modern variety can contain even as much as 1 GB or, in an increasing number of cases, even more. Furthermore, in 2009, the microcomputer has now progressed to the point of commonly incorporating (in a single chip package) two to four processing units and having other characteristics that make it more and more difficult to distinguish conceptually between the types and capabilities of large and small machines in a meaningful way that does not involve mind-numbing detail. What is certainly true is that the microcomputer found either on the desktop or on an airline tray table is now the locus of the vast proportion of all the empirical analysis that is done by economists. In almost every sense, the composite history of the electronic stored-program computer is now present in the modern personal machine.

1.3 The existing characteristics of econometric software

To this point the focus has been upon the developmental characteristics of econometric software during the past nearly 60 years. It might seem that, on the one hand, there are historical examples and, on the other, modern examples – and that the historical examples

are interesting only historically. To a degree, this is a reasonable characterization: a number of econometric software packages were created, used for a period of time, often years, and then dispensed with. However, it is not entirely correct. Although none of the packages used in the 1950s is still employed today, certain of those from the 1960s continue to be used, although in modern form. In particular, AUTOBOX, B34S, Microfit, MODLER, Mosaic, PcGive, TSP and WYSEA all began to be developed then and, in certain cases, still incorporate some original source code. Furthermore, the majority of these programs continue to be developed by their original principal developers. Others, including AREMOS, FP, IDIOM, LIMDEP, ModelEasy+, RATS, SHAZAM, and Soritec, each originally came to life on mainframes in the 1970s, as did SAS, SPSS, and other well known statistical software packages. In some cases, these too continue to be developed by their original developers. All these programs were converted to microcomputers, generally beginning at various times during the period 1980–85. In contrast, REG-X began to be developed on a Tandy microcomputer in 1979, was moved to a minicomputer, and then to the PC in the 1980s. Yet others, including EViews (as MicroTSP), Gauss, and Stata, began to be developed on the microcomputer in the 1980s, joined by Betahat, EasyReg, Ox, and the present-day incarnation of TROLL in the 1990s. Of all the existing, recognized programs, only Gretl began to be developed in the present century, albeit on the basis of 'inherited' code, although there are also certain Gauss- and Ox-based special applications that have been created during the past few years. Gauss and Ox are themselves classifiable as high-level econometric programming languages. Taken as a group, the packages just mentioned constitute the econometric software packages used by economists today. Quite clearly, the majority of them date from before 1980, so that their history and the history of the 'automatic' stored-program computer are rather extensively intertwined.

This intertwining is also indicated by these packages' collective hardware history: their development spans the cycles of hardware change since all but the earliest times. For example, the first use of the computer by economists at the University of Pennsylvania included the use of the UNIVAC, the immediate design successor to the EDVAC [52, 197], although this is arguably incidental. However, links to the second generation are firm. At least three of the existing packages began to be developed on second-generation computers, and several more on third-generation ones. The fundamental distinguishing hardware characteristic of the second-generation computer was the original introduction of the transistor, which occurred first in 1959 with the IBM 7090/94 [214]. Another machine, the IBM 7040, was effectively an IBM 7090 'lite.' The IBM 1130 and 1620, used in several cases by economists, were second-generation, small mainframes principally designed for scientific use. The CDC 6400, used in at least one case, can also be described as a second-generation mainframe machine, although it is also architecturally compatible with the earlier CDC 6600, designed by Seymour Cray, which is commonly regarded as the first supercomputer. Interactive econometric computing began in 1970 on a Digital Equipment PDP-10, a type of machine also later used by Bill Gates and Paul Allen (`www.pdpplanet.com`). The IBM 360 was a third-generation machine and in a number of cases was used by econometric software developers, as were also its successors, the IBM 370 and the 3090. Other econometric software developers, especially those in the UK, even if they did not actually cut their teeth on the first EDSAC, can nevertheless date their earliest work to the use of the Atlas, particularly the machines at the Universities of Cambridge and London, or even the EDSAC 2 or Titan [226]. More recently, econometric

software has involved the use of Apple, Tandy, the Victor 9000, the RS/6000, several Sun machines, and multiple generations of the IBM PC and compatibles. The inference to be drawn is that econometric software enjoys a long and rich hardware patrimony, one that has been only partially described here.

However, until very recently, as has been discussed, the design and development of econometric software packages has been part of the econometric deep background. Only certain individual developers have ventured into print to any significant degree [19, 63, 64, 113, 115, 160, 163, 202, 203, 208, 209, 225, 230, 244]. Furthermore, although the user guides and reference manuals commonly provided with individual programs often provide some aspects of their history, these accounts tend to be presented selectively, ordinarily without technical details. The most readily available historical description of the existing econometric software packages, albeit still somewhat limited, is found in a compendium published in 2004 [211]. This compendium comprises edited accounts by each of the current principal developers of each of the existing packages, with certain exceptions to this rule; the exceptions occur mainly in the case of historically significant programs that are no longer maintained today. Other, more selective, descriptions of particular econometric software packages available in 1983 and earlier can be found in an article of that date by Arne Drud [56], articles in a special issue of the *Journal of Economic Dynamics and Control* [130], and in minimally descriptive compilations of statistical software by Ivor Francis and others [83].

1.3.1 Software characteristics: broadening and deepening

It is said that the past is a foreign country, but if the detailed, step-by-step record is now difficult to recover, it is possible to describe the salient modern characteristics of these packages, using for reference an earlier interactive survey made in 2003. This survey was taken in conjunction with the compilation of a special volume on econometric computing, published simultaneously in the *Journal of Economic and Social Measurement* (2004, vol. 29) and as a book, which includes the compendium mentioned earlier [212]. In that context, a number of the operational characteristics of the individual packages are documented using summary charts. As a practical matter, as indicated earlier, recall that it was in the first few years of the 1960s that research-driven computer use noticeably began. At that point, the transition from using desktop calculators to the electronic computer at first occurred simply as a modal transfer: calculations previously made on the calculator began to be made instead using the computer [52, 87, 225]. After that first step came the rather slow process of incorporating into this computer use both more comprehensive data management and more than simple parameter estimation methods. The earliest recognizable econometric software, as mentioned, commonly took the form of separate, usually single purpose, programs classifiable individually as data management, data transformation, and regression programs, the latter in their original form not always easily distinguished from the ‘statistical’ programs of that day. To the degree that evident differences existed in the mid-1960s, the most obvious characteristic of econometric software was less of a tendency to include stepwise regression and more to include simultaneous equation techniques, such as limited information maximum likelihood or two-stage least squares. It was the late 1960s before programs became more than rudimentary in operating style and before econometricians even began to think about something as conceptually sophisticated as software design.

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

The particular econometric software developments during the past 50 years can be classified in various ways. Certain of these represent a widening or broadening of econometric techniques, tests, and other operations implemented in software. Others represent a capital deepening process, in the sense of more sophisticated implementations that, in some cases, took the form of more complete algorithms that subsume the capability to perform any of a multiplicity of techniques, or perhaps some combination of these simultaneously. In other cases, they took the form of combining in the same program a sequence of operations that involved a high degree of mutual integration, such as permitting parameters to be estimated as a first-stage operation, followed by the very nearly automatic creation of model equations, and then incorporating those estimates, as a next stage, leading finally to the creation of a functionally complete model, capable of being solved. Such broadening and deepening can be considered to be algorithmic in nature.

However, another aspect of this software development took the form of the creation of progressively more sophisticated interfaces. One type of interface is the human interface, functionally related to the way in which the user of a program both controls the operations performed and either perceives or comprehends the results. As discussed earlier, in the 1960s, sometimes even in the 1970s, program control was effected by choices made using numbers located in fixed fields on punched cards or paper tape. This type of control has long since been replaced by the use of the WIMP (windows, icons, menus, and pointing methods) graphical interface and even earlier by the use of free-form, if still stylized, command languages. The results obtained may, in turn, be displayed in tabular form, or as graphs, or as other perceivable objects, such as an equation or a list of equations. Comprehension, as opposed to simple perception of the program's output, obviously can be aided by interface design, even if there has been considerably less attention paid by econometric software developers to this aspect of the human interface than to enabling simple perception.

Another type of interface is the machine interface, the way in which a given computer either receives input from or sends output to one or more other machines. The idea of this interface began increasingly to be a consideration at the beginning of the 1970s, when it became feasible to connect not only one computer to another, but also individual users to machines remotely from a dumb terminal via a telecommunications link, either dial-up or dedicated. Peer-to-peer machine linkages were often difficult to achieve in those days, for computers were commonly designed to operate stand-alone, not as either intelligent or dumb correspondents; connections then generally required some type of master–slave protocol. More recently, the machine interface has of course taken the form of either a local-area network (LAN) or a wide-area network (WAN) connection, the latter including both the Internet and other machine-to-machine linkages. For econometric software developers, these have generally been behind-the-scenes innovations, since making these linkages is an operating system task. Consequently, these developers have not ordinarily been involved in the establishment of machine interconnection protocols as such. However, once such connections began to be made, remote data retrieval and database management began to become much more important environmentally [11, 12, 104, 205, 206, 213], even if today it is still most common for econometric software to be designed simply to read in data from some type of text file or an Excel or some other spreadsheet file, rather than to query a relational or other database system using SQL (Structured Query Language) or other procedural language.

Mary Morgan [182] and Qin Duo [199] have each considered carefully the process of the development of econometric theory and the way in which the ideas of Frisch, Haavelmo, and Koopmans, among others, and the work of Tinbergen, Klein, Goldberger, and others during the early days of macroeconometric model building combined to establish both econometric practice and its received theoretical support at the beginning of the 1960s. Qin's assertion (p. 65) that 'estimation can be seen as the genesis of econometrics, since finding relationships has always been the central motive and fulfilment of applied modeling activities' expresses well what can be regarded as a motivating thought behind the beginning efforts to employ the electronic computer more generally in the first few years of the 1960s. However, the operative philosophical position in those years was often that expressed in 1958 by Haavelmo [97, p. 351], that 'the most direct and perhaps most important purpose of econometrics has been the measurement of economic parameters that are only loosely specified in general economic theory.' Of course, this measurement often took place without sufficiently taking into account his clearly stated qualification (p. 352) that the quantification of economic phenomena had in the preceding 25 years appropriately come to be interpreted to extend 'not only to the measurement of parameters in would be "correct" models, but to the field of testing, more generally, the acceptability of the form of a model, whether it has the relevant variables, whether it should be linear, and many other similar problems.' The methodology debates at the end of the 1970s and into the 1980s stand as testimony to the continued lack of testing as a practice, as has been evaluated elsewhere [210].

In the early 1960s, the electronic computer, as it became progressively more commonly available, represented to economists the potential to perform computations not feasible previously. Eisenpress's creation in 1959 of a program that implemented limited information maximum likelihood was followed in 1962–63 by the efforts of Zellner and Stroud, referred to earlier, to implement the two- and three-stage least squares [257] and seemingly unrelated regression equations [256] techniques. This work marks the first

time that particular estimation techniques were both introduced in the literature [254, 258] and implemented contemporaneously in software that could be used by others. A short time after that, in 1963–64, Mike Wickens programmed full information maximum likelihood, based upon a later published formulation by James Durbin [59] that, among other things, utilized Newton–Raphson convergence and demonstrated that the second iteration of the process generated three-stage least squares estimates. Elsewhere, during this time, there were other econometricians who implemented sophisticated estimation techniques in software; much of this work took place in Canada, the UK, the USA, and New Zealand [31, 136]. In most cases, these efforts can be seen to be motivated by the desire to make these calculations specifically for the sake of it. Other efforts in the middle to late 1960s, including follow-on work in New Zealand [25, 26, 194], as well as the program development that took place at the Brookings Institution in Washington, DC [57, 58, 159], and that at the Wharton School of the University of Pennsylvania [67–69, 197, 216], represented much more the need to support the estimation, construction, and use of macroeconometric models. However, as this was the takeoff period of econometric software development, representing the first dispersed attempt to create a software infrastructure, the effect in almost all cases was generally broadening, rather than deepening, as more and more estimation and even model solution techniques became embodied in software.

A broadening also took place in the 1970s, which in many cases and in a similar way also at first represented the separate efforts of individual econometricians, yet has since resulted in the general availability of packages such as AUTOBOX, B34S, BRAP, FP, IDIOM, LIMDEP, MicroFit, PcGive, RATS, and SHAZAM. Characteristically, during the 1970s, these appear to have been developed, or in certain cases expanded, either as an individual reaction to the unavailability (or simply the local absence) of appropriate software or else to solve some perceived econometric problem. Especially in the case of MicroFit and PcGive, this work sometimes took the form of software development that over the years increasingly incorporated misspecification tests and other evaluative features. Most of this broadening, beginning then and extending to the present day, effectively took the form of the addition of econometric techniques. However, this accretion represented not simply more techniques able to be applied in a given macroeconomic time-series context, but instead, in certain cases, the development of software to be used in a different data or economic environment, such as cross-section or microeconomic. The greater availability of survey data, both cross-section and panel, as well as econometricians' advocacy of Bayesian, time-series analysis, and other specific methodologies, or new areas of investigation, such as financial theory, provided some of the initial broadening stimulus in the 1970s. In the 1980s and 1990s, the market possibilities provided by the microcomputer and, in later years, the Internet, added extra stimulus. However, some of these packages, even in their early days, broadly supported the applied research of economics departments and groups of economists at such diverse places as Auckland, the Brookings Institution, Chicago, Cambridge, Harvard, the London School of Economics, Minnesota, MIT, Pennsylvania, Princeton, and Wisconsin.

The phenomenon of software deepening is both most evident and easiest to describe in the case of programs developed for research teams associated with large-scale econometric model projects. The need to manipulate and display substantial quantities of data in conjunction with the creation and use of such models, starting in the mid-1960s, led increasingly during the 1970s to the creation of large-scale economic database management

systems, both separately and as subcomponents of such packages as EPS, FP, MODLER, Mosaic, TROLL, and XSIM [205]. More broadly, the computer and more intensive data processing generally led to the greater availability of data of all types. However, from the 1960s to the later 1980s, data series used in research often needed to be acquired in hard-copy form and then keypunched. The associated expense obviously provided an incentive to develop ways to move the data once in machine-readable form from one place to another with a minimum of effort, as well as to manipulate the observations easily. Models containing 300 or more equations only became possible because of the computer hardware and software advances that began in the 1960s, although at first models of this size strained the technology. In the early 1970s, to create a 200-equation model was commonly held to require a year's effort on the part of a team of 10–12 people [165]; in 1987, in contrast, one person working a week could estimate, compile, and successively solve a 300-equation model [43, 203]. The objects that are associated with macroeconometric models containing hundreds or even thousands of equations consist of data series, which explains the development of not only database management capabilities, but also equations, multiple tables, graphical displays, and other such items that also need to be managed effectively. In addition, there was a need to incorporate labor-saving features: the manual coding of individual equations itself was time consuming, but in addition likely to result in transcription errors. Otherwise, and in common with other types of software, deepening also took the form of the creation of program components capable of performing a variety of selected operations given a particular input stream [111]. As indicated earlier, this intensification process can be considered either as an internal program phenomenon, as just briefly described, or else in connection with the development of human command interfaces that make possible the more sophisticated control of a program's operation.

1.3.2 Software characteristics: interface development

As discussed earlier, and in greater detail in another place [203], one of the evolutionary characteristics of econometric software was the early development of explicit econometric modeling languages, which began in the late 1960s. The employment here of the term 'language' refers to the command structure as a human interface, which permits the user of this type of software to describe to the software the operations to be performed using an algebraic syntax and vocabulary, together with keywords and variable names – for example, resulting in transformation commands (and identities) such as

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

Here, the variable names (Y, C, I, G, X, M) have an obvious mnemonic quality, and as command elements each constitutes a symbolic reference to a vector of observations, inasmuch as the use of the program's command language not only directly invokes the retrieval of observations from an organized database (and perhaps subsequently the storage of results there) but also defines and causes the calculations and other operations (written in the language) that are associated with the construction, maintenance, and use of an econometric model that might contain even hundreds or a thousand or more equations. Once created, such a program can also be used more prosaically to make simple data transformations, as shown above, as well as to perform regressions, to execute a variety of analytical tasks, and to display tables, graphs, and the like, all in a

comparatively user-friendly way. Consequently, as previously described, the development of econometric modeling languages in the 1970s was often associated with the formation of economic consulting and forecasting firms, which then made available to a wider public both software services and economic data for analysis [206].

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

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

During the 1970s, there was almost always a noticeable difference between the human interface of the econometric software packages typically used by academic economists and that experienced mainly by business and other non-academic economists, who used the econometric modeling language type of interface just described. This difference in part reflects that, during the 1970s, batch processing mainframes and minicomputers were much more commonly available in academic environments than were computers with time-sharing operating systems. The typical self-programming academic econometrician in the 1970s might, in any case, have had little incentive to develop a sophisticated language interface for a program, compared to the incentive to focus upon econometrically interesting algorithms, but in a card (or paper tape) oriented batch environment there was

even less reason. AUTOREG, B34S, LIMDEP, and most other such programs were originally developed with an algorithmic focus, rather than the interface. As indicated, programs such as Damsel, EPS, MODLER, and XSIM were more human interface biased in their development. The combination of differences in developer incentives and their environments explain the particular diverse characteristics and almost bipolar orientation of econometric software development during the 1970s.

However, it is also pertinent that, until about 1978, much of the design and development of econometric software occurred under relatively isolated conditions. As indicated, there was a time in the early 1970s that journals, in particular *Econometrica*, appeared ready to publish articles and notes about software, but for whatever reason this was a short-lived, Prague spring. With certain exceptions [19, 63, 64], it was only at the end of this decade that program descriptions and algorithmic details noticeably began to appear in the disciplinary literature [51, 115, 130, 132, 151, 188, 227]. Otherwise, econometric software and its documentation ordinarily passed from hand to hand, even if user guides to statistical programs had begun to appear in university bookstores. In the days before microcomputers, software purchases were commonly made organizationally, usually by people who worked in computer centers and spoke of 'statistical,' rather than 'econometric,' software; in addition, it was decidedly uncommon for software of any type to be prominently marketed at economic association and society meetings. Even as late as 1983, computational methods were ordinarily considered separately from any explicit consideration of their algorithmic computer implementation, and the citations that appeared in the formal economics and econometrics literature were often not directly related to any such implementation [200], a practice not unknown even today.

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

However, in 1985, a new type of software began to become available, the earliest example familiar to economists being Gauss [119]. It is possible to argue that too sharp a distinction has just been made, that the econometric modeling languages already offered capabilities similar to those of these new packages, albeit described in the depths of thick manuals, but it is useful to ignore this particular fine point in order to focus on the difference in orientation of these two types of econometric software package. Packages such as AREMOS, MODLER, and XSIM are examples of econometric modeling languages (EML) [203], as has been described, but Gauss, Ox, and possibly other similar packages are effectively econometric programming languages (EPL). The critical difference is the object the user works with: an econometric modeling language characteristically has as its

objects specific, well defined econometric techniques, to include estimators with explicit names. Other objects take the form of time-series variables, model equations, and models, but also a range of variable transformations, defined in terms of algebraic and arithmetic operators, and, as well, also implicit functions.

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

Suffice it to say that these are fuzzy sets. However, ignoring such classification complexities, econometric software can today be categorized into various types. First are the standard estimation packages that provide an economist with the ability to perform a given set of econometrically defined operations, operations that are specifically defined by the software developer. Notice that the operational characteristic in this case consists of the user selecting from a set of options, possibly using a menu. There is next a mid-range, which most obviously includes the econometric modeling languages, with the characteristic that the economist is required not only to make certain selections but also to control how particular operations are performed: he or she must form equations, combining variables and operators and possibly implicit functions, and thereby build a model. These models can be solved or simulated. The results can be plotted or produced as tabular displays. Advanced estimation packages (AEP) or generalized estimation packages (GEP) that offer a selection of choices and incorporate a macro language capability should also be included in this classification, as offering a subset of capabilities and features. Finally, the econometric programming language in turn offers less in the way of prefabricated statistical and mathematical objects, but more scope to create new econometric, statistical, and mathematical forms. It might also be possible to infer that an econometric programming language is most suited for use by econometricians, as creators of emerging techniques, as opposed to applied economists, who are more likely to use established methodologies, hence another type of package. Obviously, these sharp distinctions are most meaningful when considering polar examples of these package types.

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

to offer much, even if this capacity might also imply the need to specify explicitly the calculations made in each case.

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

All this sounds good, but there are certain aspects of econometric software that make these concepts less than immediately applicable. In the first place, in the case of econometric software, there is no necessity for there to be a direct correspondence between what appears on the screen and the computations that are made. Users of this software generally do not and will not know the algorithmic details of the computations performed, for the simple reason that individual developers do not ordinarily publish these details. Furthermore, whatever the user specifies in the form of a command, there is no necessary relationship between this command and the calculations performed by the software. At issue here 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, or the user's freedom to set initial values and other control parameters [172, 173]. However, fundamentally, whatever the user provides as a set of commands will be interpreted by the software package used, acting as an intermediating agent. The actual calculations then performed will be determined and controlled by the software developer. Furthermore, at least in certain cases, it can be argued that it is desirable that the user of the package not be allowed to control precisely how the program does what it does: the user cannot be presumed to be a knowledgeable numerical analyst nor a skilled programmer.

1.3.3 Directives versus constructive commands

In certain respects, the discussion has come full circle since the introductory section of this chapter. Implicitly it was argued there that, over the past 30–40 years, specialization has occurred, with economists in effect ceding responsibility for the design and development of econometric software to a minority of econometricians. One of the implications of the modern development of this software – namely, the creation of econometric programming languages – would appear on the face of it to provide any economist (once again?) with the capability, in effect, to design and develop his or her own software, but now in a way that avoids the complexities, yet achieves the goal of allowing that economist to determine the constructive characteristics of whatever applied econometric research

project he or she might wish to imagine. The one objection that has been made to this idea is the argument just posed that, in any case, the designer and developer of any econometric programming language used is essentially able to remain in complete control as an intermediating agent. The normative question that naturally arises is this: To what degree should the designer/developer exert this control?

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

dependent: CE,
regressors: YPD, CELAG1;

in another:

$$\text{CE} = F(\text{YPD},\ \text{CE}(-1));$$

or, in a third:

$$\text{reg. command:}\quad \text{CE} = \text{c1} * \text{YPD} + \text{c2} * \text{CE}(-1) + \text{c3}.$$

All three directive types are found in the command languages of existing econometric software packages. It is also true that, in some cases, pulldown or dropdown menus will be used in order to identify, progressively, the dependent and regressor variables.

All directive forms are constructively equivalent, inasmuch as, by definition, none does anything except direct that a certain type of operation be performed. In each case, the command's meaning and the particular corresponding default operation will have been established by the program's designer; that is, the meaning of the directive is completely established by the syntax and vocabulary of the program used. However, as illustrated, in some cases a directive can have constructive features, either seemingly or actually; for example, in the second command above, the term CE(−1) obviously constitutes the directive that the variable named CE is to be retrieved from the program's data storage component and then, constructively, lagged by one period before the observations on this variable are used as one of the regressors in the implied regression. Alternatively, in the third case, the directive would seem to indicate constructively that the regression is to be linear in parameters. Nevertheless, notice that none of these directives is linked automatically to a specific estimator, except by default.

In contrast, a textbook consideration of the so-called general linear model and ordinary least squares regression will commonly begin with a statement like: 'Consider the linear specification

$$y = X\beta + u,$$

where $y \in \Re^T$ are the observations on the dependent variable, $X \in \Re^{T \times k}$ are the observations on k regressor variables, $\beta \in \Re^k$ are the unobserved constant parameters, and $u \in \Re^T$ are the unobserved disturbances.'

Constructively, the most direct approach to performing such a regression is to compute the sums of squares and cross-products of all the relevant variables and then load them

into a matrix. Conventionally – as shown in almost any modern econometrics textbook [46, 94, 129], and even in many older ones [88, 128, 236] – this matrix can be formed so that the cross-products of the dependent variable with the regressor variables border those of the regressor variables alone, producing the matrix:

$$\begin{pmatrix} X'X & X'y \\ y'X & y'y \end{pmatrix}.$$

The next procedural step is simply to invert the interior matrix, $X'X$.

The specific way that this is done can matter constructively. In particular, if the inversion process is programmed so that, as it takes place, the operations are extended to the rightmost column of the original bordered matrix, this inversion can be performed in a way [91] that causes that column simultaneously to become the estimated values of the parameters, denoted by b:

$$\begin{pmatrix} (X'X)^{-1} & b \\ y'X & y'y \end{pmatrix},$$

where, of course, b is constructively defined (in textbooks) as:

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

At first glance, it might appear that the same result is obtained whichever of these two possible routes is taken to obtain the estimates, but inasmuch as few, if any, program command languages are designed to permit the user to choose between them, if a user is allowed to specify the estimator constructively, the estimates are almost certain to be obtained in the manner implied by the textbook definition of b. After all, what is the user's alternative, in practice? That is, other than by programming at the most fundamental level, how is the user to specify constructively the precise algorithms involved, not only in this simple case but generally? Equally determinant, where in the econometrics textbook literature will he or she discover the alternative algorithms that might be used?

However, when programming, the extra matrix multiplications implied by the textbook definition of b are not computationally efficient if carried out explicitly: most importantly, they can result in additional rounding error, compared to the simultaneous generation of the inverse and the parameter estimates. In this and other important cases, an almost inevitable consequence is that the textbook mathematical statement of an econometric calculation will not necessarily directly correspond to the best algorithmic statement of its computational logic. The result can be an operational divergence in both the statistical and numerical properties of the econometric calculations [232]. However, in many cases (although not all), the possibility of this divergence can be traced to the general circumstance that the finite memory space that can be allocated for the representation of any given number inevitably allows only a finite set of the real numbers to be represented by the computer without approximation, leading to approximation error even before the first calculation is made. Incidentally, with reference to these calculations, please note that this discussion ignores many details that should be examined were the complete story to be told, including truncation errors that occur when representing derivatives and integrals finitely and other possible formulation-related approximation errors [201].

It is important to understand why the property of finiteness is important and, as well, that memory space is inevitably limited by competing claims. It is not difficult

to appreciate why number representation space limitations were necessarily severe for the first electronic computers, with as little as 1 kB of total memory; clearly, the ability then to create anything approaching even a minimally useful program implied serious restrictions. However, the effective space gain implied by the growth of random access memory (RAM) memory to 32 kB, 64 kB, or even 640 kB during the next 40 years was less than might appear at first sight, once account is taken of the concomitant increase in the amount of work done by the typical program, especially one designed to be used by many different people. In the early 1960s, for whatever particular reason, the space used to represent a given real number was commonly limited by default to 32 binary digits, which allowed a floating-point representation of only five or six decimal digits of precision for the fractional part. This so-called single-precision standard, although capable of being increased, has continued to be a common default. One of the reasons is that only in the 21st century has the RAM available on the typical computer increased to the point that it has become possible to imagine writing comprehensive computer programs without the need to consider carefully offsetting storage space issues each step of the way. Yet, even today, the properties of computers are still limited by decisions made in 1980, or even before – at a time when 1 GB of RAM per user seemed prospectively stupendous. However, even ignoring these history-determined restrictions, it is not reasonable to suppose that any time in the near future that programs can be written without taking some account of space restrictions. A given computer might have 1 GB or even 2 GB of RAM, but overall memory limitations still exist. They exist as a consequence of a continuing battle for space among programs, as programs have become more capable, space demanding, and numerous, not to mention the need to consider such things as execution speed requirements. Even the operating system, possibly incorporating an Internet browser, is a program that competes with others for space or at least operation time, which is functionally much the same thing in a virtual machine world. Similarly, each program's interface code competes for space with its analytically oriented code. Likewise, the number of observations on each economic variable competes for space against the number of variables that can be operated on at any one time.

Any floating-point number essentially consists of two space-taking parts, a fractional part and an exponent. Quite clearly, it is always possible in principle to choose an exponent that will map any real number into the open set $(-1, 1)$ to provide a decimal fraction as the fractional part value, which then removes the need to cope with the decimal point itself. For example, consider the number 3345 647, which can be stated, in terms of the fractional part, as 0.334 565 if it is represented to six places. The exponent is obviously 10^7. Except as a result of a programming error, an economist is unlikely to encounter an observation in his or her work the scale of which is greater than the machine's default limiting size for exponents, especially as the modern microcomputer, properly utilized, actually supports the IEEE numeric standard. However, the number of digits of precision available to represent the fractional part is necessarily limited, and this problem arises irrespective of whether one considers single-precision numbers, double-precision numbers, or for that matter numbers of any finite-precision number n. In addition, also reflecting this finiteness, error can be introduced by the need to convert between the decimal numbers that are input as data and the base 2 values used internally by the machine. If n is finite, approximation error is inevitably involved.

In the case of linear ordinary least squares parameter estimates, or linear parameter estimates more generally, the fundamental lurking danger is the degree to which the data

used might be highly collinear. The problem is the ease with which, in this event, the calculations can result in computed values that have no meaningful precision whatsoever, possibly resulting from intermediate values that are effectively random numbers. To monitor this circumstance, the condition number, defined as the ratio of the largest to the smallest singular values of the matrix to be inverted, can be computed and displayed [17, 22]. In the presence of high collinearity, the calculation of the sums of squares and cross-products matrix can produce values that are affected by rounding and truncation errors, which are exacerbated in particular by the process of matrix inversion, so that the accuracy of the final results – parameter estimates and the derived misspecification statistics – becomes highly sensitive to the input values [100, 232]. A specific manifestation is what is known as 'catastrophic number cancellation,' which can occur when one large number is subtracted from another, both of which numbers agree in their n leading digits, leading to an extreme loss of significant digits and error propagation. For a more detailed discussion of this problem, see, for instance, Golub and Van Loan [90], particularly chapter 2, or Householder [123], chapter 1. Another problem posed by finitely precise number representation is that comparisons can only discriminate approximately between numbers: for floating-point real values x and y, it is not meaningful to ask if $x = y$ exactly, but only if $\|x - y\| \leq \epsilon$, where ϵ is some suitably chosen small number. One of the implications is that, even in the linear case, computations such as the inversion of a matrix must be carried out with due regard for the effect of the data used, as a matter of conditioning, as well as the fact that in the end the solution is always approximate rather than exact [118, 228].

To the mathematical economist, there is ordinarily a sharp conceptual difference between a linear and a nonlinear problem. To the economist as a numerical analyst, the computational difference between a linear and a nonlinear problem is essentially that the latter requires additional calculations, with each successive iteration typically introducing additional rounding and approximation errors – although, of course, nonlinear problems can additionally involve other specific computational issues [173]. Curiously enough, error propagation is not necessarily likely to be particularly great in the case of multiplication, division, or taking square roots, but simply adding operands of different sign can, in extreme cases, lead to catastrophic cancellation [228, pp. 11–12]. All this can be considered in much greater detail, but the main point is that truly precise calculations do not occur within an electronic computer – the name of the game is the minimization of calculation error, not its absolute elimination. A classic study of the computational problem of error accumulation is that by Harold Hotelling [122], but see also Wilkinson [250], Belsley *et al*. [22], and most recently McCullough [170] and Stokes [229, 232]. For a consideration of the algorithmic properties of simultaneous equation estimators, see for example Kontoghiorghes *et al*. [80, 142, 145].

Longley's 1967 study [155] of the numerical accuracy of regression programs first brought to the attention of econometric software developers and others the problem of rounding error in the context of single-precision floating-point numbers. There has since been a significant amount of study of appropriate numeric methods, most recently by Stokes [232], who also addresses data storage precision and alternative matrix inversion techniques. The methods for matrix inversion are determined by the characteristics of the data: in particular, near-singular matrices should use the QR or singular-value decompositions applied directly to the data matrix, rather than the usual Cholesky factorization of $X'X$. However, it is often not evident in advance just how collinear the data are. At

one time, this uncertainty created a conundrum; for instance, in the 1960s, in the case of mainframes, and the 1980s, in the case of microcomputers, there were concerns about the demands placed upon the capabilities of existing CPUs by highly accurate matrix inversion techniques. Today, in contrast, although still depending of course upon the size of the matrix, most situations can be taken in one's stride inasmuch as most of the estimation problems that the economist is likely to encounter day to day will involve small to moderate-sized matrices that are unlikely to tax the latest computers.

The omission of textbook treatments and considerations of these issues provides an important reason why command-line directives, rather than constructive commands, should predominate. However, in addition to the knowledge requirement that constructive commands pose, there is also what might be called the problem of computational completeness, alluded to earlier. Textbook presentations provide only partial coverage of the relevant range of formulas and calculations. Programming ordinary least squares, or any other parameter estimation method, in the most generally reliable manner requires information difficult to find in the literature. For example, how should one treat special cases, such as the suppression of the intercept, which, among other things, affects the validity of certain diagnostic statistics that are ordinarily displayed with the parameter estimates? Or, how should the program react if the user chooses to regress a variable on itself or omits entirely all regressor variables? How should one treat missing observations, either at the extremes or in the interior of the range? These events are not necessarily the result of sensible actions by the program user, but rather are actions that, if not dealt with gracefully, can cause the program to crash, or possibly, without warning, produce incorrect or inappropriate results. Ideally, any action that the user chooses to make should produce a meaningful response by the program, either an intelligible error message or a sensibly performed action.

There is also the more general, not yet fully explored, question of the characteristics of the supplementary statistics, alluded to earlier. An existing, recently discovered problem with current econometric software packages is that they often present supplementary statistics that are eclectically named; for instance, one program's 'White test' is another's 'Breusch–Godfrey' [210]. Or one program might display a particular statistic in log form that is presented by another in level form. Often, each of these values numerically implies the other, once their relationship is understood, but the user is not necessarily told which is being presented. Or one program might omit the 'inessential constant' from the calculation of a statistic, whereas another includes it. The following question therefore arises: What nomenclature should be adopted for a particular set of supplementary statistics so as to make them instantly intelligible to the reasonably knowledgeable user, and, beyond this, what specific values should be presented in each case, assuming the underlying calculations have indeed been accurately made? Although in certain cases some values have been found to be miscalculated, usually, at least in the case of ordinary least squares (OLS), the problem is one of presentation, not calculation. However, so far, except for certain special cases [34, 172], only OLS-related statistics have been evaluated and, of course, OLS is just one estimation method, so that this judgment is provisional. It is pertinent to note that, except in rare instances, such as the footnote on p. 269 of the 2003 edition of Greene's textbook [94], such issues have not been addressed in modern econometrics textbooks, seemingly leaving readers the freedom to assume that they can blithely self-evaluate econometric software simply on the basis of its interface modernity or other subjective criterion.

Additionally, there is the matter of which set of supplementary statistics a program should provide. Although it is possible to identify a common set of supplementary statistics that are generally provided as basic by existing econometric packages, eclecticism reigns beyond this basic set. Many programs allow the user to select from additional supplementary statistics, among those they are programmed to calculate optionally. There is therefore the open question: Should the average applied economist be presumed to know which statistics to choose to display, or should a program's designer make a default selection in advance? Should there also be provision for the selection to change, based upon the particulars of the situation? Which statistics are mutually compatible and which are potentially contradictory? Which statistics are better in the case of small samples? And what is a small sample? These are questions that are in part design questions, and in part transcendental questions that may have no real answer, at least at present, but they are nevertheless questions that must be answered, well or badly, during each package's design process.

Some existing packages allow their users to generate still more supplementary statistics, or even additional parameter estimators, among a variety of other options. Indeed, if any program simply permits the values of the residuals and the 'predicted' values of the dependent variable to be saved, the knowledgeable user can create numerous additional misspecification test statistics, the primary question then being the ease with which this can be done. The design issue is only in part that of which facilities are consciously provided, even if the choices made in advance by a program's designer cannot help but limit or promote the subsequent choices of its users. However, whichever particular design choices have been made and implemented, the effect in the aggregate will be to affect how applied economics research is performed. For almost any product on the open market, there are always some 'after market' options that become available, but most products are used 'as is.' In practice, few programs are radically reconfigured during use, even when written to permit this. What may be critical here is the degree to which programs are designed to be user enabling, for those who create econometric software packages do not in fact exercise final control: it is in principle possible for any economist to create anew his or her own econometric software package from the ground up. It might therefore be argued that packages should be created for those who are themselves most unlikely to program – but the implications of such an argument need to be carefully considered before it is made in earnest.

1.3.4 Econometric software design implications

It is evident from this limited discussion that there are many detailed questions of econometric software design that have yet to be either satisfactorily answered or even widely admitted to need careful consideration. However, there are nonetheless at least two possible general precepts of econometric computing. The first is that it involves the development of algorithms that permit the calculations to be made that are specified by econometric theory, where these are sufficiently broadly defined so as to include all the operations that are described in the economics literature. Of course, that literature includes what has been referred to earlier as the mainstream economic and econometric journals, which have a practical importance and relevance to economists. At the same time, there is a more broadly defined literature, equally respectable, that may include, in addition or instead, journals published by the Association for Computing Machinery, statistical

organizations around the world, and others in the context of which the design of econometric software might well be seen as being more universally defined. In this context, there will be certain computations and operations that are regarded as being specifically econometric in nature, but these may not be algorithmically unique nor in themselves uniquely econometric. In particular, an atomistic view of the computations performed by an econometric software package, in terms of such operations as matrix inversion, the solution of a set of nonlinear equations, and the calculations that underlie the production of misspecification statistics, are each quite able to be interpreted as subject matter to be considered in the context of this other, ostensibly non-econometric, literature. Should econometric software in fact be defined by the data used, in line with the suggestion made by Granger, quoted earlier? Should it be defined instead by the particular set of calculations that are performed, recognizing that this is an especially fuzzy set?

Recall from the earlier discussion the properties of the data 'that are not considered in standard statistics texts or are not sufficiently emphasized there for economists.' It is not altogether a churlish question to ask where in the econometrics literature one goes to find a discussion of the particular properties of economic data. It is certainly evident where one might go to find a discussion of the general properties of time-series data, and of the cointegration and other tests that might be applied to particular examples to attempt to discover their specific statistical properties, but it is not obvious how this information should best be utilized to design a software package that makes these tests particularly applicable to economic data, especially in a way that enables the effective testing of any given dataset. Are the properties of economic data merely statistical? Similarly, it is clear from what has been said so far that it is broadly possible to identify something called 'econometric software,' but at the same time it is also clear that there is not necessarily a checklist that can be constructed that uniquely defines its properties, either of solely its own software type or in comparison with the properties of some other type of software, such as 'statistical software.'

It is tempting to suggest that the absolutely indispensable requirement of econometric software is the provision for parameter estimation based upon regression techniques, but of course there are many statistical software packages that also include regression as a technique. As an alternative, it is possible to assert that what distinguishes econometric software is the provision of parameter estimation techniques that provide for the estimation of the parameters of simultaneous equations, but if this idea is carried too far the final result would be to exclude software packages that are commonly used by economists 'as if' econometric software packages. What is evident in all of this is that the econometrics literature and econometrics software do not necessarily make a particularly well matched couple. The collective characteristics of the existing econometric software packages actually extend much more broadly than does the subject coverage of the mainstream econometrics literature, either the textbooks or the journals literature. Furthermore, although there is likely to be a present feeling on the part of econometric theorists that it is econometric theory that defines the appropriate operative content of econometric software packages, in fact what is beginning to occur is that the software is beginning to define the frontiers of econometric theory.

Of course, it is incorrect to say that this is beginning to occur. It began to occur some years ago, but the collective capacity of these software packages to define both operational econometrics and the frontiers of econometric theory is only just on the verge of becoming widely recognized. In this regard it might be possible to argue that what is happening,

perhaps to the horror of the most fastidious, is that theory is being overrun by practice, but the simple truth is that the development of an operative econometric software package requires that choices be made. Those choices, for good or ill, then become econometric practice. In some particular cases, such as in the case of PcGive, the bit has actually been taken between the teeth. From the perspective of this program's principal designer, David Hendry, the software has become the teaching agent. Hendry openly asserts that 'we see our PcGive books as textbooks with a strong computational bent (and use them that way), rather than manuals' [109]. The effect is to present the user of this software with the particular set of techniques that are offered by the software, thus excluding from consideration any techniques not offered. But, at the same time, as textbooks, the PcGive guidebooks provide a direct pedagogic mechanism, thus marrying econometric theory with operative econometrics. More generally, it is only to the degree that the developers of econometric software severally and collectively incorporate particular techniques that the average applied economist becomes able to use them. How these techniques are implemented, and which techniques are implemented, shape applied economic research.

Yet the whole story has not so far been told. In the early days, econometric software packages were designed so as to offer their users a choice from among a given collection of techniques and facilities, even if the user might be at liberty to recode. In contrast, today, considering polar cases, it is possible to characterize econometric software either as offering the user a choice among a selected set of techniques, often using a WIMP interface, or else as a high-level language that is defined in terms of matrices, loops, and other programming language constructs and that permits the user to specify the calculations to be performed. Growing out of the development of the econometric modeling languages in the 1970s, but of course also affected by the general development of computer software everywhere, including operating systems and computer programming languages, the provision of a high-level language interface and programming language constructs widened the scope and increased the flexibility of applications software for both econometric software and other end-user programs. Of these polar examples, the first type provides the user with a minimal actual need to confront the particular computational characteristics of the computer, but at the cost of accepting an intermediated choice of input and output conventions, as well as techniques. The second ostensibly provides the user with the capability to control the computational environment, but at the same time does not impose the responsibility to deal directly with the computer's actual operating system.

The word 'ostensibly' is used here as a reminder, to indicate that the user may actually have little or no control over the precise way in which calculations are performed. As discussed earlier, in reality, whatever commands he or she issues are always interpreted by the program and then translated into the operational commands actually obeyed by the program. It is a misunderstanding of the context to think of any software package as necessarily freely enabling the user. Fundamentally, on the one hand, there is absolutely no point in such intermediation if the user actually takes on the developer's freedom and responsibility: he or she may as well start from scratch. On the other hand, if the user of a particular program is not able by using it to ignore the various numerical analytical issues that define best practice, why use an applications program at all? This tension between accepting intermediation and writing a program from scratch began in 1957 when Fortran relieved the users of that day of the need to program in machine or assembly language. It is likely to continue in the foreseeable future.

It is an interesting question how the economist is affected by the various developer solutions to the three earlier identified research problems of data acquisition and management, user interface, and the particular econometric techniques the existing packages make available. There are at least two classes of users of econometric software. One of these consists of academic economists, who tend to use small datasets that tend to be fixed for extended periods of time, essentially during a particular research project. These economists appear to tend to place a premium on the particularity of the techniques offered and may also prefer to be seen to be using noticeably 'cutting edge' techniques. The topical 'mix' that these economists represent may also be changing over time, with financial theory progressively becoming more represented. The other class consists of business economists (or at least non-academic economists), who characteristically are often responsible for monitoring and analyzing the performance of one or more economic entities or economies. These economists may use either small or large datasets, but seldom those that are fixed over time. They often have similar training to academic economists, but may not be quite as sensitive to the need to be seen to be using the academically most fashionable techniques. Instead, they may place a premium on ease of use and upon the ability to acquire data and produce results rapidly. They may be less sensitive to the modernity of the interface. The desires of these two classes of users, among others, create the dynamic for the future development of econometric software, and it may also be possible to see the existing packages as reflecting the specific historical demands of these groups or, it has been argued [166-169, 171, 175, 208], even an indifference to software that is both accurate and carefully designed.

The classification of econometric software developers is equally interesting. One characteristic, commonly unrecognized, is that those econometricians who develop this software, as a group, include a relatively high proportion of econometrics textbook writers, or if not textbooks specifically, then books on econometric methodology. Furthermore, although not always in a formalized way, there has additionally been some tendency in recent years for particular econometrics textbooks to become tied in some fashion to particular econometric software packages. In other cases, as indicated earlier, there is a perceptible tendency for the manuals of particular packages to take on at least some the characteristics of textbooks, not always unintentionally. Furthermore, in certain cases, econometrician developers are editors of economics or econometrics journals, or else serve on the advisory boards of journals or are otherwise linked in some fashion to particular journals. Overall, there is a strong tendency for at least one of the people closely associated with each econometric software package to hold an academic post, possibly reflecting the need for a day job if one happens to develop econometric software, which may also signal the condition of the market for this software. In short, even though it is generally true that the development of econometric software is a path neither to riches nor to academic tenure, it is wrong to think of this activity as being an intensely commercial, academically unrelated business.

However, it also needs to be recognized that economists, and econometricians, do not use only 'econometric software,' even apart from Word and Excel. Econometrics, as it is portrayed in textbooks, may be principally concerned with parameter estimation and related issues, and may tend not to include formally such computational issues as the solution of simultaneous equation models, even in the case of those models estimated using standard econometric techniques. But, as has been discussed above, such limitations do not necessarily impose software restrictions. Articles published in journals such

as *Computational Economics* and the *Journal of Economic Dynamics and Control* that involve the use of software will often involve the use of Matlab, Mathematica, R, or S-Plus, which originate outside the discipline of economics, lack econometric estimation facilities as such, although these can be self-programmed using the facilities of these languages, and therefore often can solve systems of equations that can be identified as econometric models [18, 20, 133]. Certain types of models, such as computable general equilibrium (CGE), may involve the use of other software packages, like GAMS (General Algebraic Modeling System), which is currently classified as being 'specifically designed for modeling linear, nonlinear and mixed integer optimization problems,' that fit within the classification of quantitative economics software or operations research, but not econometric software strictly. What have been called calibrated macroeconomic models (CMM), and particularly the real business cycle models of Kydland and Prescott, may use unidentified software of a proprietary nature. The aforementioned reticence of economists to identify publicly the specific software used in their research makes it difficult to be certain about all the details.

These software packages have capabilities in common with econometric modeling language software, such as AREMOS, MODLER, TROLL, WYSEA or WinSolve, or with econometric programming languages, such as GAUSS or Ox, to the extent that, if the distinguishing capabilities are defined algorithmically, there may not be any greater degree of functional difference between 'non-econometric' and econometric software packages than between econometric modeling packages, econometric programming languages, and other econometric software packages. This is particularly the case if account is taken of the fact that the code to estimate parameters is actually a small proportion of the code of packages such as AREMOS, MODLER, or TROLL. There are particular procedures used by macroeconometric model builders, such as what are called type 1 and type 2 'fixes' or 'ragged edge' solutions, that may not be provided by Matlab or GAMS, for instance, that serve therefore as distinguishing features, but apart from these differences, at least certain algorithmic characteristics may be quite similar. Because of such classification difficulties, the recently published compendium [211] of econometric software was based upon the way in which software developers self-classified their software, not just properties capable of being objectively defined. This approach is not a perfect solution to the problem of classification, but it does permit inferences to be drawn about econometric software following from developers' stated intentions. Furthermore, over time, packages designer-identified as econometric are more likely to characterize econometric practice than packages such as Matlab or S-Plus. Of course, as suggested earlier, to the degree that economists use the latter type of package, econometric practice may be broadened by the implied non-disciplinary influence.

1.4 Conclusion

This chapter is intended to convey that econometricians can take pride in the progress to date, but, at the same time, that there is a need for a better and more general appreciation of the process of software development and of the relationship of this to the future development of econometrics. A possible natural comparison lies in the contrast between physics and engineering, or even pure and applied mathematics. That these are false comparisons is, however, easily seen. Physics as a set of abstract principles is not affected by the degree

to which physical structures are built, nor are the principles of mathematics affected by any application, even if it is also true that as disciplines both physics and mathematics are inevitably human, thus worldly, endeavors. In contrast, as a subdiscipline of economics, econometrics exists as the direct result of original attempts to explain the observed. Its subject matter is expressly defined by this history. Anyone who takes the time to mark the difference between the early econometrics textbooks of Klein, Tinbergen, and Tintner, through the mid-years of Christ, Goldberger, and Johnston, to the present day cannot but be struck by the degree to which what have been perceived as real-world problems have conditioned the evolution of the subject. For example, the work of Weiner and Kalman, Box and Jenkins, although perhaps adopted originally somewhat out of context by econometricians, at inception fundamentally represented attempts to explain respectively signal propagation, machine operation, and the implications of autoregressive processes. Similarly, the work of Engle, leading to the examination of (generalized) autoregressive conditional heteroskedasticity (ARCH and GARCH) phenomena, both in its application to financial theoretic problems and for its inspiration, depends upon an original real-world context. The work of Granger, Klein, Sargan, Stone, Theil, Tinbergen, Zellner, and others could also be considered in this light. Item by item, including the relative shift from the macroeconomic estimation methods emphasis of the early textbooks to both the modern emphasis on diagnostic tests and the consideration of methods related to microeconomic cross-section or panel data applications, what has been driving the development of econometrics has been empirical observation, albeit sometimes at a slight remove.

With these thoughts in mind, what is evident from the historical literature, in the form of well known articles by Leamer [152], Lucas and Sargent [156, 157], and Sims [221], among others [92, 103], is the possibility that at least some of the heat and noise of the methodology debates of the 1970s and 1980s reflected a lack of reality grounding, a degree of misperception of the actual computational realities of the 1960s and 1970s, as compared to the inflated computational expectations that arose during that period [66]. Even if defensibly based on a visceral concern for the applied econometric evaluation process in those days, a concern that remains, possibly in a revised form, even today, much of that methodology discussion arguably lacked suitable supportive specificity, in at least some cases representing what has recently been described as an excess of 'theory without measurement' [66] that then gave rise to a distaste for confronting the hard realities too closely. It may be indicative that, recently, Granger, in his *Econometric Theory* interview [193], expresses certain misgivings about the 'whole evaluation process – both in econometrics and in economics generally. I want to know how people evaluate the theory and how people evaluate a technique and how to value the model.' He goes on to observe that 'a lot of literature is losing the viewpoint that we are here to learn about the actual economy,' and furthermore that at least some economists are '... playing games when they write papers.' Later (p. 62) he reflects that he 'would like to see much more use of relevant economic theory in model building' and notes that [he is] 'worried that the Econometrics Society includes both economic theorists and econometricians and that in their meetings the two sides never talk – or virtually never talk. There are few joint sessions involving both theorists and econometricians.' In such an environment, it has been easy for the profession to ignore such seemingly mundane matters as the design of econometric software and its impact and to push all considerations of such issues into the deep background. Yet Granger's words suggest the underlying importance of a better general understanding of the computational process.

Acknowledgments

I am very grateful for the advice, comments, suggestions, and corrections that I have received from a number of readers, particularly Jerry Adams, Richard Anderson, David Belsley, Allin Cottrell, Robert Dixon, Tom Doan, Clive Granger, Bruce McCullough, Houston Stokes, and Arnold Zellner. The opinions expressed are, however, entirely my own. I am also wholly responsible for all remaining errors of fact and omission.

References

1. Announcement, 1972. NBER Computer Center for Economics and Management Science. Annals of Economic and Social Measurement, 1(1): 103.
2. Anon., 1948. Proceedings of a Symposium on Large-Scale Digital Calculating Machinery. Cambridge, MA: Harvard University Press.
3. Adams, M.O., 1986. The Kentucky Economic Information System. In Exemplary Systems in Government Awards '85–'86: the State of the Art. Urban and Regional Information Systems Association. pp. 144–158.
4. Adams, M.O., 1981. 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. Medford, NJ: Learned Information.
5. Adams, M.O. and F.E. Ross, 1983. The energy data base within the Kentucky Economic Information System. Review of Public Data Use, 11(1): 75–78.
6. Adelman, F. and I. Adelman, 1959. The dynamic properties of the Klein–Goldberger model. Econometrica, 27: 596–625.
7. Adelman, I., 2007. The research for the paper on the dynamics of the Klein–Goldberger model. Journal of Economic and Social Measurement, 32(1): 29–33.
8. Aitchison, J. and J.A.C. Brown, 1957. The LogNormal Distribution. Cambridge: Cambridge University Press.
9. Alexander, L.A. and T.B. Jabine, 1980. Access to Social Security microdata files for research and statistical purposes. Review of Public Data Use, 8: 203–224.
10. Altman, M., J. Gill, and M.P. McDonald, 2004. Numerical Issues in Statistical Computing for the Social Scientist. Hoboken, NJ: John Wiley & Sons, Inc.
11. Anderson, R.G., 2006. Replicability, real-time data, and the science of economic research: FRED, ALFRED, and VDC. Federal Reserve Bank of St. Louis Review, pp. 81–93.
12. Anderson, R.G., W.H. Greene, B.D. McCullough, and H.D. Vinod, 2008. The role of data and code archives in the future of economic research. Journal of Economic Methodology, 15(1): 99–119.
13. Ball, R.J. and S. Holly, 1991. Macroeconometric model-building in the United Kingdom. In A History of Macroeconometric Model-Building, R.G. Bodkin, L.R. Klein, and K. Marwah, Editors. Aldershot: Edward Elgar, pp. 195–230.
14. Barker, T.S., W. Dada, and W. Peterson, 2004. Software developments in the Cambridge Growth Project 1960–1976: the origins of software for space–time economics. Journal of Economic and Social Measurement, 29: 173–182.
15. Begg, I. and S.G.B. Henry, 1998. Applied Economics and Public Policy. Cambridge: Cambridge University Press.
16. Belsley, D.A., 1986. Centering, the constant, first-differencing, and assessing conditioning. In Model Reliability. Cambridge, MA: MIT Press, pp. 117–152.

17. Belsley, D.A., 1991. Conditioning Diagnostics. New York: John Wiley & Sons, Inc.
18. Belsley, D.A., 1996. Doing Monte Carlo studies with Mathematica. In Computational Economics and Finance. Modeling and Analysis with Mathematica, H.R. Varian, Editor. New York: Springer, pp. 300–343.
19. Belsley, D.A., 1974. Estimation of systems of simultaneous equations, and computational specification of GREMLIN. Annals of Economic and Social Measurement, 3: 551–614.
20. Belsley, D.A., 1999. Mathematica as an environment for doing economics and econometrics. Computational Economics, 14(1–2): 69–87.
21. Belsley, D.A. and E. Kuh, Editors, 1986. Model Reliability. Cambridge, MA: MIT Press.
22. Belsley, D.A., E. Kuh, and R.E. Welsch, 1980. Regression Diagnostics. New York: Wiley-Interscience.
23. Berger, R.L., 2007. Nonstandard operator precedence in Excel. Computational Statistics and Data Analysis, 51: 2788–2791.
24. Bergmann, B.R. and R.L. Bennett, 1984. Macroeconomic models on microfoundations: data requirements. Review of Public Data Use, 12(2): 91–96.
25. Bergstrom, A.R., 1967. The Construction and Use of Economic Models. London: English Universities Press.
26. Bergstrom, A.R., 1967. Selected economic models and their analysis. New York: American Elsevier.
27. Bergstrom, A.R., K.B. Nowman, and C. Wymer, 1992. Gaussian estimation of a second order continuous time macroeconometric model of the United Kingdom. Economic Modelling, pp. 313–351.
28. Bergstrom, A.R. and P.C.E. Phillips, 1993. Models, methods, and applications of econometrics: essays in honor of A.R. Bergstrom. Cambridge, MA: Blackwell.
29. Berndt, E.R. *et al*., 1974. Estimation and inference in nonlinear structural models. Annals of Economic and Social Measurement, 3: 653–665.
30. Bodkin, R.G., 1999. 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. New York: Physica, pp. 41–60.
31. Bodkin, R.G., L.R. Klein, and K. Marwah, 1991. A History of Macroeconometric Model-Building. Brookfield, VT: Edward Elgar.
32. Boutillier, M. and B. Durand, 1986. Investigations in the causal structure of the yearly OFCE model. Journal of Economic Dynamics and Control, 10: 131–137.
33. Bracy, D. *et al*., Editors, 1975. Computer Programs Available for Distribution. Philadelphia, PA: Wharton Econometric Forecasting Associates.
34. Brooks, C., S. Burke, and G. Persand, 2001. Benchmarks and the accuracy of GARCH model estimation. International Journal of Forecasting, 17: 45–56.
35. Brown, B.W., 1975. WEFA batch solution program, WEFA program usage guide #2. In Computer Programs Available for Distribution, D. Bracy *et al*., Editors. Philadelphia, PA: Wharton Econometric Forecasting Associates.
36. Brown, J.A.C., H.S. Houthakker, and S.J. Prais, 1953. Electronic computation in economic statistics. Journal of the American Statistical Association, 48(263): 414–428.
37. Brundy, J.M. and D.W. Jorgenson, 1974. The relative efficiency of instrumental variables estimation of systems of simultaneous equations. Annals of Economic and Social Measurement, 3(4): 679–700.
38. Bruno, G. and R. De Bonis, 2004. A comparative study of alternative econometric packages with an application to Italian deposit interest rates. Journal of Economic and Social Measurement, 29: 271–296.

39. Byte, 1984. Guide to the IBM Personal Computers. Byte: The Small Systems Journal, 9(9).
40. Byte, 1986. Inside the IBM PCs. Byte, The Small Systems Journal, 11(11).
41. Campbell-Kelly, M. and M.R. Williams, Editors, 1985. The Moore School Lectures. Cambridge, MA: MIT Press.
42. Christ, C.F., 1966. Econometric models and methods. New York: John Wiley & Sons, Inc.
43. Cooper, F., 1987. Modeling the U.S. trade sector. Paper presented to the 1987 MODLER Users Conference, October 21, Philadelphia, PA.
44. Cramer, M., 2006. The working conditions of econometric scholars, then and now. Statistica Neerlandica, 60(2): 194–2005.
45. Czamanski, S., 1968. An econometric model of Nova Scotia. Halifax, NS: Dalhousie University Institute of Public Affairs.
46. Davidson, J., 2000. Econometric Theory. Oxford: Blackwell.
47. Deardorff, A.V. and R.M. Stern, 1978. What Have We Learned From Linked Econometric Models? A Comparison of Fiscal-Policy Simulations. Stockholm: Institute for International Economic Studies, University of Stockholm.
48. Dennis, J.E. and R.E. Welsch, 1978. Techniques for nonlinear least squares and robust regression. Communications in Statistics, 7: 345–359.
49. Dennis, J.E.J., D.M. Gay, and R.E. Welsch, 1981. An adaptive nonlinear least-squares algorithm. ACM Transactions on Mathematical Software, 7: 348–368.
50. Dennis, J.E.J., D.M. Gay, and R.E. Welsch, 1981. Algorithm 573 NL2SOL–an adaptive nonlinear least-squares algorithm. ACM Transactions on Mathematical Software, 7: 369–383.
51. Dent, W.T., 1980. Computation in econometric models. Journal of Econometrics: Annals of Applied Econometrics, 12.
52. Desai, M., 2007. Memories of Monroe: econometric computing in the early 1960s. Journal of Economic and Social Measurement, 32(1): 35–38.
53. Drud, A., 1983. Interfacing modelling systems and solution algorithms. Journal of Economic Dynamics and Control, 5: 131–149.
54. Drud, A., 1976. Methods for Control of Complex Dynamic Systems: Illustrated by Econometric Models. Lyngby: IMSOR.
55. Drud, A., 1977. An Optimization Code for Nonlinear Econometric Models Based on Sparse Matrix Techniques and Reduced Gradients. Lyngby: Matematisk Institut Danmarks Tekniske Højskole.
56. Drud, A., 1983. A survey of model representations and simulation algorithms in some existing modeling systems. Journal of Economic Dynamics and Control, 5: 5–35.
57. Duesenberry, J.S. *et al*., 1969. The Brookings Model: Some Further Results. Chicago, IL: Rand McNally.
58. Duesenberry, J.S. *et al*., 1965. The Brookings Quarterly Econometric Model of the United States. Chicago, IL: Rand McNally.
59. Durbin, J., 1988. Maximum likelihood estimation of the parameters of a system of simultaneous regression equations. Econometric Theory, 4: 159–170.
60. Eisenpress, H., 1959. Forecasting by Generalized Regression Methods. Limited-Information Estimation Procedure. IBM 704 Program IB LI. New York: International Business Machines Corporation, Data Systems Division.
61. Eisenpress, H., 1962. Note on the computation of full-information maximum-likelihood estimates of coefficients of a simultaneous system. Econometrica, 30(2): 343–348.
62. Eisenpress, H. and J. Greenstadt, 1966. The estimation of nonlinear econometric systems. Econometrica, 34(4): 851–861.

63. Eisner, M., 1972. TROLL/1–an interactive computer system for econometric research. Annals of Economic and Social Measurement, 1: 95–96.
64. Eisner, M. and R.S. Pindyck, 1973. A generalized approach to estimation as implemented in the TROLL/1 system. Annals of Economic and Social Measurement, 2: 29–51.
65. Eliasson, G., 1978. A Micro-to-Macro Model of the Swedish Economy. Stockholm: Almqvist & Wiksell.
66. Ericsson, N.R. and J.S. Irons, 1995. The Lucas critique in practice: theory without measurement. In Macroeconometrics: Developments, Tensions, and Prospects, K.D. Hoover, Editor. Boston, MA: Kluwer, pp. 263–311.
67. Evans, M.K., 1969. Macroeconomic Activity: Theory, Forecasting, and Control; an Econometric Approach. New York: Harper & Row.
68. Evans, M.K. and L.R. Klein, 1967. The Wharton Econometric Forecasting Model. Programmed by George R. Schink. Philadelphia, PA: Economics Research Unit, University of Pennsylvania.
69. Evans, M.K. and L.R. Klein, 1968. The Wharton Econometric Forecasting Model. Programmed by George R. Schink, Second, Enlarged Edition. Philadelphia, PA: Economics Research Unit, University of Pennsylvania.
70. Fair, R.C., 1977. A note on the computation of the Tobit estimator. Econometrica, 45(7): 1723–1727.
71. Fair, R.C., 1974. On the robust estimation of econometric models. Annals of Economic and Social Measurement, 3(4): 667–678.
72. Fair, R.C., 1971. A Short-Run Forecasting Model of the United States Economy. Lexington, MA: Lexington Books, D.C. Heath.
73. Fair, R.C., 1984. Specification, Estimation, and Analysis of Macroeconometric Models. Cambridge, MA: Harvard University Press.
74. Fair, R.C., 1994. Specification, Estimation, and Analysis of Macroeconometric Models, Second Edition. Cambridge, MA: Harvard University Press.
75. Fair, R.C. and W.F. Parke, 1993. The Fair–Parke Program for the Estimation and Analysis of Nonlinear Econometric Models. New Haven, CT: Yale University Press.
76. Fair, R.C. and J.B. Taylor, 1983. Solution and maximum likelihood estimation of dynamic nonlinear rational expectations models. Econometrica, 51: 1169–1185.
77. Farrell, M.J., 1957. The measurement of productive efficiency. Journal of the Royal Statistical Society, Series A (General), 120(III): 253–290.
78. Fisher, P.G., S. Holly, and A.J. Hughes Hallett, 1986. Efficient solution techniques for dynamic non-linear rational expectations models. Journal of Economic Dynamics and Control, 10: 139–145.
79. Fisher, P.G. and A.J. Hughes Hallett, 1988. Iterative techniques for solving simultaneous equations systems: a view from the economics literature. Journal of Computational and Applied Mathematics, 24: 241–255.
80. Foschi, P., D.A. Belsley, and E.J. Kontoghiorghes, 2003. A comparative study of algorithms for solving seemingly unrelated regressions models. Computational Statistics and Data Analysis, 44: 3–35.
81. Foschi, P. and E.J. Kontoghiorghes, 2003. Estimating seemingly unrelated regression models with vector autoregressive disturbances. Journal of Economic Dynamics and Control, 28: 27–44.
82. Foschi, P. and E.J. Kontoghiorghes, 2003. Estimation of VAR models: computational aspects. Computational Economics, 21: 3–22.

83. Francis, I., Editor, 1981. Statistical Software: A Comparative Review. New York: North Holland.
84. Frisch, R., 1933. Editorial. Econometrica, 1(1): 1–4.
85. Gambardella, A. and B.H. Hall, 2004. Propriety vs. public domain licensing of software and research products. European University Institute, Department of Economics, EUI Working Paper ECO No. 2004/15.
86. Gatu, C. and E.J. Kontoghiorghes, 2006. Estimating all possible SUR models with permuted exogenous data matrices derived from a VAR process. Journal of Economic Dynamics and Control, 30: 721–739.
87. Goldberger, A.S., 2004. Econometric computation by hand. Journal of Economic and Social Measurement, 29: 115–118.
88. Goldberger, A.S., 1964. Econometric Theory. New York: John Wiley & Sons, Inc.
89. Goldberger, A.S. and V. Hofer, 1962. A users guide to the multiple regression program RGR. Systems Formulation and Methodology Workshop, Paper 6207. Madison, WI: Social Systems Research Institute, University of Wisconsin.
90. Golub, G.H. and C.F. Van Loan, 1996. Matrix Computations, Third Edition. Baltimore, MD: Johns Hopkins University Press.
91. Goodnight, J.H., 1979. A tutorial on the sweep operator. American Statistician, 33: 149–158.
92. Granger, C.W.J., 1999. Empirical Modeling in Economics: Specification and Evaluation. Cambridge: Cambridge University Press.
93. Granger, C.W.J., 1994. A review of some recent textbooks of econometrics. Journal of Economic Literature, 32: 115–122.
94. Greene, W.H., 2003. Econometric Analysis, Fifth Edition. Upper Saddle River, NJ: Prentice-Hall.
95. Greene, W.H., 1980. Maximum likelihood estimation of econometric frontier functions. Journal of Econometrics, 13: 27–56.
96. Greene, W.H., 1980. On the asymptotic bias of the ordinary least squares estimator of the tobit model. Econometrica, 48: 505–514.
97. Haavelmo, T., 1958. The role of the econometrician in the advancement of economic theory. Econometrica, 26(3): 351–357.
98. Hall, B.H., 2003. Email to C.G. Renfro, re. the History of econometric software development.
99. Hall, B.H., 2003. Email to C.G. Renfro, re. TSP and symbolic processing.
100. Hall, R.E., 1967. The Calculation of Ordinary Least Square. Working Paper #2, Department of Economics, Massachusetts Institute of Technology.
101. Hall, R.E., 2003. Email to C.G. Renfro and B.H. Hall, re. History of econometric software development.
102. Hall, R.E., 1967. Matrix Operations in Econometrics. Working Paper #1, Department of Economics, Massachusetts Institute of Technology.
103. Hall, S., 1995. Macroeconomics and a bit more reality. Economic Journal, 105: 974–988.
104. Harrison, T. and C.G. Renfro, 2004. TSX and TSE: a proposal for an extended time series data interchange and transfer format. Journal of Economic and Social Measurement, 28: 339–358.
105. Harvey, A.C., 1990. The Econometric Analysis of Time Series, Second Edition. LSE Handbooks in Economics. Cambridge, MA: MIT Press.
106. Hausman, J., 1975. An instrumental variable approach to full information estimators for linear and certain nonlinear econometric models. Economica, 43(4): 727–738.

107. Hausman, J.A., 1974. Full information instrumental variables estimation of simultaneous equation systems. Annals of Economic and Social Measurement, 3(4): 641–652.

108. Haveman, R.H. and K. Hollenbeck, 1980. Microeconomic Simulation Models for Public Policy Analysis. New York: Academic Press.

109. Hendry, D.F., 2003. Email to C.G. Renfro, re. Final version of software survey paper.

110. Hendry, D.F., 2003. J. Denis Sargan and the origins of the LSE econometric methodology. Econometric Theory, 19: 457–480.

111. Hendry, D.F., 1976. The structure of simultaneous equation estimators. Journal of Econometrics, 4: 51–88.

112. Hendry, D.F. and J.A. Doornik, 1999. Empirical Econometric Modelling Using PcGive, Volume I. London: Timberlake Consultants Ltd.

113. Hendry, D.F. and J.A. Doornik, 1999. The impact of computational tools on time-series econometrics. In Information Technology and Scholarship, T. Coppock, Editor. London: British Academy, pp. 257–269.

114. Hendry, D.F. and H.-M. Krolzig, 2003. The Properties of Automatic 'Gets' Modelling. Economics Papers 2003-W14, Economics Group, Nuffield College, University of Oxford.

115. Hendry, D.F. and F. Srba, 1980. AUTOREG: a computer program library for dynamic econometric models with autoregressive errors. Journal of Econometrics, 12: 85–102.

116. Herbert, R.D., 2004. Modelling programming languages–appropriate tools? Journal of Economic and Social Measurement, 29: 321–338.

117. Hickman, B.G., L.R. Klein, and Project LINK, 1998. LINK Proceedings, 1991, 1992: Selected Papers from Meetings in Moscow, 1991, and Ankara, 1992. Singapore: World Scientific.

118. Higham, N., 2002. Accuracy and Stability of Numeric Algorithms. Second Edition. Philadelphia, PA: SIAM.

119. Hill, R.C., 1989. Learning Econometrics Using Gauss. New York: John Wiley & Sons, Inc.

120. Holland, P.W. and R.E. Welsch, 1977. Robust regression using iteratively reweighted least-squares. Communications in Statistics, 6: 813–827.

121. Hollinger, P., 1996. The Stacked-time simulator in TROLL: a robust algorithm for solving forward-looking models. Paper presented at the Second International Conference on Computing in Economics and Finance, Geneva, Switzerland, June 26–28.

122. Hotelling, H., 1943. Some new methods in matrix calculation. Annals of Mathematical Statistics, 14(1): 1–34.

123. Householder, A.S., 2006. Principles of Numerical Analysis. Mineola, NY: Dover.

124. Houthakker, H.S., 1951. Some calculations on electricity consumption in Great Britain. Journal of the Royal Statistical Society, Series A, 114(III): 351–371.

125. Hughes Hallett, A.J., 1981. Some extensions and comparisons in the theory of Gauss–Seidel iterative techniques for solving large equations systems. In Proceedings of the 1979 Econometric Society Meeting: Essays in Honour of Stefan Valavanis. E.G. Charatsis, Editor. Amsterdam: North-Holland.

126. Hughes Hallett, A.J. and L. Piscitelli, 1998. Simple reordering techniques for expanding the convergence radius of first-order iterative techniques. Journal of Economic Dynamics and Control, 22: 1319–1333.

127. Intriligator, M.D., 1978. Econometric Models, Techniques, and Applications. Englewood Cliffs, NJ: Prentice-Hall.

128. Johnston, J., 1963. Econometric Methods. New York: McGraw-Hill.

129. Johnston, J. and J. DiNardo, 1997. Econometric Methods, Fourth Edition. New York: McGraw-Hill.

130. Kendrick, D. and A. Meeraus, 1983. Special issue on modeling languages. Journal of Economic Dynamics and Control, 5(1).

131. Kim, K.-S. *et al*., 1979. Time Series Processor: A Computer Language for Econometrics and Statistics Interfaced With KDI Data Base, Revised Edition. Seoul, Korea: Korea Development Institute.

132. Kirsch, J., 1979. TROLL Primer, Third Edition. Cambridge, MA: MIT Center for Computational Research in Economics and Management Science.

133. Kleiber, C. and A. Zeileis, 2008. Applied Econometrics With R. New York: Springer.

134. Klein, L.R., 1950. Economic Fluctuations in the United States 1921–1941. Cowles Commission Monograph, Vol. 11. New York: John Wiley & Sons, Inc.

135. Klein, L.R., 1946. A post-mortem on transition predictions of national product. Journal of Political Economy, 54(4): 289–308.

136. Klein, L.R., 1960. Single equation vs. equation systems methods of estimation in economics. Econometrica, 28: 866–871.

137. Klein, L.R. and M. Dutta, 1995. Economics, Econometrics and the LINK: Essays in Honor of Lawrence R. Klein. Contributions to Economic Analysis, no. 226. Amsterdam: Elsevier.

138. Klein, L.R. and A.S. Goldberger, 1955. An econometric model of the United States, 1929–1952. Contributions to Economic Analysis, no. 9. Amsterdam: North-Holland.

139. Knuth, D.E., 1986. The IBM 650: an appreciation from the field. Annals of the History of Computing, 8(1): 50–54.

140. Knuth, D.E., 1970. Von Neumann's first computer program. ACM Computing Surveys, 2(4): 247–260.

141. Kontoghiorghes, E.J., 2004. Computational methods for modifying seemingly unrelated regressions models. Journal of Computational and Applied Mathematics, 162: 247–261.

142. Kontoghiorghes, E.J., 2000. Inconsistencies in SURE models: computational aspects. Computational Economics, 16: 63–70.

143. Kontoghiorghes, E.J., 2000. Parallel strategies for solving SURE models with variance inequalities and positivity of correlation constraints. Computational Economics, 15: 89–106.

144. Kontoghiorghes, E.J. and M.R.B. Clarke, 1995. An alternative approach for the numerical solution of seemingly unrelated regression equations models. Computational Statistics and Data Analysis, 19: 369–377.

145. Kontoghiorghes, E.J. and E. Dinenis, 1997. Computing 3SLS solutions of simultaneous equation models with a possible singular variance–covariance matrix. Computational Economics, 10: 231–250.

146. Krolzig, H.-M. and D.F. Hendry, 2001. Computer automation of general-to-specific model selection procedures. Journal of Economic Dynamics and Control, 25: 831–866.

147. Kuh, E., 1974. Introduction. Annals of Economic and Social Measurement, 3(4): i–iii.

148. Kuh, E., 1972. Some notes on the research program at the NBER Computer Research Center for Economics and Management Science. Annals of Economic and Social Measurement, 1(2): 233–236.

149. Kuh, E. and J. Neese, 1982. Econometric model diagnostics. In Evaluating the Reliability of Macro-Economic Models, G.C. Chow and P. Corsi, Editors. New York: John Wiley & Sons, Inc.

150. Kuh, E. and R.E. Welsch, 1980. 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. Washington, DC: National Bureau of Standards, pp. 445–475.

151. Lane, T., Editor, 1979. TROLL Reference Manual (Standard System), Second Edition. Cambridge, MA: MIT Center for Computational Research in Economics and Management Science, MIT Information Processing Services.

152. Leamer, E.E., 1983. Let's take the con out of econometrics. American Economic Review, 73: 31–43.

153. Leontief, W.W., 1948. Computational problems arising in connection with economic analysis of industrial relationships. In Proceedings of a Symposium on Large-Scale Digital Calculating Machinery. Cambridge, MA: Harvard University Press.

154. Lilien, D., 2000. Econometric software reliability and nonlinear estimation in EViews: comment. Journal of Applied Econometrics, 15(1): 107–110.

155. Longley, J.W., 1967. An appraisal of least squares programs for the electronic computer from the point of view of the user. Journal of the American Statistical Association, 62: 819–841.

156. Lucas, R.E. and T.J. Sargent, 1981. Rational Expectations and Econometric Practice. Minneapolis, MN: University of Minnesota Press.

157. Lucas, R.E.J. and T.J. Sargent, 1981. After Keynesian macroeconomics. In Rational Expectations and Econometric Practice, R.E.J. Lucas and T.J. Sargent, Editors. Minneapolis, MN: University of Minnesota Press, pp. 295–319.

158. Malinvaud, E., 1966. Statistical Methods of Econometrics. Translated by A. Silvey. Chicago, IL: Rand McNally.

159. McCarthy, M.D., 1992. The Cowles Commission, the Brookings Project, and the econometric services industry: successes and possible new directions: a personal view. Econometric Theory, 8: 383–401.

160. McCracken, M.C., 1967. A computer system for econometric research. Social Science Information, 5: 151–158.

161. McCracken, M.C., 1967. Data administration in an information system. In Conference on Government Information Systems. Ottawa: Economic Council of Canada.

162. McCracken, M.C., 1966. DATABANK System: Users' Manual. Ottawa: Economic Council of Canada.

163. McCracken, M.C. and K. May, 2004. Issues in the development of early solution facilities for very large macroeconometric models. Journal of Economic and Social Measurement, 29: 167–172.

164. McCracken, M.C. and C.A. Sonnen, 1973. MASSAGER User's Manual. Ottawa: Statistics Canada.

165. McCracken, M.C. and C.A. Sonnen, 1972. A system for large econometric models: management, estimation, and simulation. In Proceedings of the Association for Computing Machinery Annual Conference, Boston, Mass., August 1972. New York: Association for Computing Machinery.

166. McCullough, B.D., 2000. The accuracy of Mathematica 4.0 as a statistical package. Computational Statistics, 15: 279–299.

167. McCullough, B.D., 1999. Econometric software reliability: EViews, LimDep, Shazam, and TSP. Journal of Applied Econometrics, 14(2): 191–202.

168. McCullough, B.D., 2000. Is it safe to assume that software is accurate? International Journal of Forecasting, 16: 349–357.

169. McCullough, B.D., 1997. A review of RATS v4.2: benchmarking numeric accuracy. Journal of Applied Econometrics, 12: 181–190.

170. McCullough, B.D., 2004. Wilkinson's tests and econometric software. Journal of Economic and Social Measurement, 29: 261–270.

171. McCullough, B.D., K.A. McGeary, and T.D. Harrison, 2006. Lessons From the Archive(s) at the Fed. St. Louis Review (and Other Journals). Philadelphia, PA: Drexel University.

172. McCullough, B.D. and C.G. Renfro, 1998. Benchmarks and software standards: a case study of GARCH procedures. Journal of Economic and Social Measurement, 25: 59–71.

173. McCullough, B.D. and C.G. Renfro, 2000. Some numerical aspects of nonlinear estimation. Journal of Economic and Social Measurement, 26: 63–77.

174. McCullough, B.D., C.G. Renfro, and H.H. Stokes, 2006. Re-examining 50-year-old OLS estimates of the Klein–Goldberger model. Statistica Neerlandica, 60(3): 181–193.

175. McCullough, B.D. and H.D. Vinod, 1999. The numerical reliability of econometric software. Journal of Economic Literature, 37(2): 633–665.

176. McCullough, B.D. and B. Wilson, 1999. On the accuracy of statistical procedures in Microsoft Excel 97. Computational Statistics and Data Analysis, 31: 27–37.

177. McCullough, B.D. and B. Wilson, 2002. On the accuracy of statistical procedures in Microsoft Excel 2000 and Excel XP. Computational Statistics and Data Analysis, 40: 713–721.

178. McCullough, B.D. and B. Wilson, 2005. On the accuracy of statistical procedures in Microsoft Excel 2003. Computational Statistics and Data Analysis, 49(4): 1244–1252.

179. Meeraus, A., 1983. An algebraic approach to modeling. Journal of Economic Dynamics and Control, 5: 81–108.

180. Mendelssohn, R.C., 1980. The new on-line BLS database and information system. In Readings in Business and Economic Research Management: Execution and Enterprise, J.D. Fischer, Editor. Madison, WI: Office of Research Administration, School of Business, University of Wisconsin, pp. 86–89.

181. Mizon, G.E., 1984. The encompassing approach in econometrics. In Econometrics and Quantitative Economics, D.F. Hendry and K.F. Wallis, Editors. Oxford: Blackwell, pp. 135–172.

182. Morgan, M.S., 1990. The History of Econometric Ideas. Historical Perspectives on Modern Economics. Cambridge: Cambridge University Press.

183. Nakamura, Y. and J.T. Yap, 1990. ASEAN link: an econometric study. White Plains, NY: Longman.

184. Norman, A.L., L.S. Lasdon, and J.K. Hsin, 1983. A comparison of methods for solving and optimizing a large non-linear econometric model. Journal of Economic Dynamics and Control, 6(1/2): 3–24.

185. Orcutt, G.H., 1962. Microanalytic models of the United States economy: need and development. American Economic Review, 52: 229–240.

186. Pesaran, M.H. and B. Pesaran, 1987. Microfit. A Interactive Econometric Software Package. User Manual. Oxford: Oxford University Press.

187. Pesaran, M.H. and B. Pesaran, 1997. Working with MicroFit 4.0. Interactive Econometric Analysis. Oxford: Oxford University Press.

188. Pesaran, M.H. and L.J. Slater, 1980. Dynamic Regression: Theory and Algorithms. Chichester: Ellis Horwood.

189. Peterson, W., 1987. Computer software for a large econometric model. In The Cambridge Multisectoral Dynamic Model of the British Economy, T.S. Barker and W. Peterson, Editors. Cambridge: Cambridge University Press, pp. 105–121.

190. Peterson, W., 2004. Email to C.G. Renfro, re. Econometric computing in Cambridge.

191. Peterson, W., T. Barker, and R. van der Ploeg, 1983. Software support for multisectoral dynamic models of national economies. Journal of Economic Dynamics and Control, 5(1): 81–108.

192. Phillips, P.C.B., 1988. The ET interview: Professor A.R. Bergstrom. Econometric Theory, 4: 301–328.

193. Phillips, P.C.B., 2001. The ET interview: Professor Clive Granger. In Essays in Econometrics. Collected Papers of Clive W.J. Granger. E. Ghysels, N.R. Swanson, and M.W. Watson, Editors. New York: Cambridge University Press, pp. 28–81.

194. Phillips, P.C.B. and V. Hall, 2004. Early development of econometric software at the University of Auckland. Journal of Economic and Social Measurement, 29: 127–134.

195. Prais, S.J. and H.S. Houthakker, 1955. The Analysis of Family Budgets. Cambridge: Cambridge University Press.

196. Press, S.J., 1980. Bayesian computer programs. In Bayesian Analysis in Econometrics and Statistics: Essays in Honor of Harold Jeffreys, A. Zellner, Editor. Amsterdam: North-Holland, pp. 429–442.

197. Preston, R.S., 2006. Personal Reflections on the Historical Development and Use of Computer Software Needed to Undertake Econometric Research at the Economics Research Unit, University of Pennsylvania–1960 to 1969.

198. Project LINK and J.A. Sawyer, 1979. Modelling the international transmission mechanism: applications and extensions of the Project LINK system. Contributions to Economic Analysis, no. 121. New York: North-Holland.

199. Qin Duo, 1993. The Formation of Econometrics: A Historical Perspective. Oxford: Clarendon Press.

200. Quandt, R.E., 1983. Computational problems and methods. In Handbook of Econometrics, Z. Griliches and M.D. Intriligator, Editors. Amsterdam: North-Holland, pp. 699–764.

201. Ralston, A. and P. Rabinowitz, 2001. A First Course in Numerical Analysis. Mineola, NY: Dover.

202. Renfro, C.G., 1981. The computer as a factor of production in economic data provision and research. Presented at the American Economic Association Annual Meeting, Washington, DC, 30 December 1981.

203. Renfro, C.G., 2004. The early development of econometric modeling languages. Journal of Economic and Social Measurement, 29: 145–166.

204. Renfro, C.G., 2004. Econometric software: the first fifty years in perspective. Journal of Economic and Social Measurement, 29(1–3): 9–108.

205. Renfro, C.G., 1997. Economic data base systems: further reflections on the state of the art. Journal of Economic and Social Measurement, 23(1): 43–85.

206. Renfro, C.G., 1980. Economic data base systems: some reflections on the state of the art. Review of Public Data Use, 8(3): 121–140.

207. Renfro, C.G., 1970. An Interactive Regression Program for the PDP-10. Description and User Manual. Washington, DC: Brookings Quarterly Model Project, Brookings Institution.

208. Renfro, C.G., 1997. Normative considerations in the development of a software package for econometric estimation. Journal of Economic and Social Measurement, 23(4): 277–330.

209. Renfro, C.G., 1996. On the development of econometric modeling languages: MODLER and its first twenty-five years. Journal of Economic and Social Measurement, 22(4): 241–311.

210. Renfro, C.G., 2009. The Practice of Econometric Theory. New York: Springer.

211. Renfro, C.G., Editor, 2004. A compendium of existing econometric software packages. Journal of Economic and Social Measurement, 29: 359–402.

212. Renfro, C.G., Editor, 2004. Computational Econometrics: Its Impact on the Development of Quantitative Economics. Amsterdam: IOS Press.

213. Ritter, J.A., 2000. Feeding the national accounts. Federal Reserve Bank of St. Louis Review, 82(2): 11–20.

214. Rosen, S., 1969. Electronic computers: a historical survey. ACM Computing Surveys, 1(1): 7–36.

215. Sadowsky, G., 1977. MASH: A Computer System for Microanalytic Simulation for Policy Exploration. Washington, DC: Urban Institute.

216. Schink, G.R., 2004. Simulation with large econometric models: the quest for a solution. Originally a paper presented to the Association for Computing Machinery meetings, Denver, Colorado, 12 June 1970. Journal of Economic and Social Measurement, 29: 135–144.

217. Shiller, R., 1973. A distributed lag estimator derived from smoothness priors. Econometrica, 41(4): 775–788

218. Shiskin, J. and H. Eisenpress, 1957. Seasonal adjustments by electronic computer methods. Journal of the American Statistical Association, 52: 415–449.

219. Silk, J., 1996. Systems estimation: a comparison of SAS, SHAZAM, and TSP. Journal of Applied Econometrics, 11(4): 437–450.

220. Simon, S.D. and J.P. Lesage, 1988. Benchmarking numerical accuracy of statistical algorithms. Computational Statistics and Data Analysis, 7: 197–209.

221. Sims, C.A., 1980. Macroeconomics and reality. Econometrica, 48: 1–48.

222. Slater, L.J., 1967. Fortran Programs for Economists. Department of Applied Economics Occasional Paper 13. Cambridge: Cambridge University Press.

223. Slater, L.J., 1972. More Fortran Programs for Economists. Department of Applied Economics Occasional Paper 34. Cambridge: Cambridge University Press.

224. Slater, L.J., 2004. Recollections on early days in the Cambridge Computing Laboratory. Journal of Economic and Social Measurement, 29: 119–122.

225. Slater, L.J., 1962. Regression analysis. The Computer Journal, 4(4): 287–291.

226. Slater, L.J. and T.S. Barker, 1967. Regression Program for Titan. Cambridge: Department of Applied Economics, University of Cambridge.

227. Society for Economic Dynamics and Control, 1981. News item: Third Conference of the Society for Economic Dynamics and Control (Program). Journal of Economic Dynamics and Control, 3: 393–395.

228. Stoer, J. and R. Bulirsch, 1993. Introduction to Numerical Analysis, Second Edition. New York: Springer.

229. Stokes, H.H., 2008. A benchmark-free approach to assess the accuracy of an OLS model calculation in an econometrically uncertain world. Journal of Economic and Social Measurement, 33: 1–17.

230. Stokes, H.H., 2004. The evolution of econometric software design: a developer's view. Journal of Economic and Social Measurement, 29: 205–259.

231. Stokes, H.H., 2004. On the advantage of using two or more econometric software systems to solve the same problem. Journal of Economic and Social Measurement, 29: 307–320.

232. Stokes, H.H., 2005 The sensitivity of econometric results to alternative implementations of least squares. Journal of Economic and Social Measurement, 30(1): 9–38.

233. Stokes, H.H., 1991. Specifying and Diagnostically Testing Econometric Models. New York: Quorum Books.

234. Stokes, H.H., 1997. Specifying and Diagnostically Testing Econometric Models, Second Edition. New York: Quorum Books.

235. Summers, R., 1965. A capital intensive approach to the small sample properties of various simultaneous equation estimators. Econometrica, 33(1): 1–41.

236. Theil, H., 1971. Principles of Econometrics. New York: John Wiley & Sons, Inc.

237. Trimble, G.R., 1986. The IBM 650 magnetic drum calculator. Annals of the History of Computing, 8(1): 20–29.

238. Uebe, G., G. Huber, and J. Fischer, 1988. Macro-Econometric Models–An International Bibliography. Aldershot: Gower.

239. Velleman, P.F. and R.E. Welsch, 1976. On evaluating interactive statistical program packages. Communications in Statistics, Part B–Simulation and Computation, 5: 197–208.

240. Wallis, K.F. *et al*., 1984. Models of the UK Economy. Oxford: Oxford University Press.

241. Wallis, K.F. *et al*., 1985. Models of the UK Economy: A Second Review by the ESRC Macroeconomic Modelling Bureau. Oxford: Oxford University Press.

242. Wallis, K.F. *et al*., 1986. Models of the UK Economy: A Third Review by the ESRC Macroeconomic Modelling Bureau. Oxford: Oxford University Press.

243. Wallis, K.F. *et al*., 1987. Models of the UK Economy: A Fourth Review by the ESRC Macroeconomic Modelling Bureau. Oxford: Oxford University Press.

244. White, K.J., 1978. A general computer for econometric models–SHAZAM. Econometrica, 46(1): 239–240.

245. White, K.J., L.T.M. Bui, and E.R. Berndt, 1991. The Practice of Econometrics: A Computer Handbook Using SHAZAM. Reading, MA: Addison-Wesley.

246. White, K.J., S.A. Haun, and D.J. Gow, 1988. Introduction to the Theory and Practice of Econometrics–A Computer Handbook Using SHAZAM and SAS. New York: John Wiley & Sons, Inc. For use with G.G. Judge, R.C. Hill, W.E. Griffiths, H. Lütkepohl, and T.C. Lee, 1988. Introduction to the Theory and Practice of Econometrics, Second Edition. New York: John Wiley & Sons, Inc.

247. Wickens, M., 2003. Email to C.G. Renfro, re. Early computing at the LSE.

248. Wilkes, M.V., 1956. Automatic Digital Computers. London: Methuen.

249. Wilkes, M.V. and W. Renwick, 1949. The EDSAC, an electronic calculating machine. Journal of Scientific Instruments, 26: 385–391.

250. Wilkinson, J.H., 1961. Error analysis of direct methods of matrix inversion. Journal of the Association for Computing Machinery, 8(3): 281–330.

251. Woytinsky, W.S., 1947. What was wrong in forecasts of postwar depression. Journal of Political Economy, 55(2): 142–151.

252. Zeileis, A., 2006. Implementing a class of structural change tests: an econometric computing approach. Computational Statistics and Data Analysis, 50: 2987–3008.

253. Zeileis, A. and C. Kleiber, 2005. Validating multiple structural change models–a case study. Journal of Applied Econometrics, 20: 685–690.

254. Zellner, A., 1962. An efficient method of estimating seemingly unrelated regressions and tests for aggregation bias. Journal of the American Statistical Association, 57: 348–368.

255. Zellner, A., 1989. PCBRAP: a PC Bayesian regression program. American Statistician, 43(1): 124.

256. Zellner, A., A. Stroud, and L.C. Chau, 1963. Program for Computing Seemingly Unrelated Regressions Estimates. Madison, WI: Social Systems Research Institute, Department of Economics.

257. Zellner, A., A. Stroud, and L.C. Chau, 1963. Program for Computing Two and Three Stage Least Squares and Associated Statistics. Madison, WI: Social Systems Research Institute, Department of Economics.

258. Zellner, A. and H. Theil, 1962. Three stage least squares: simultaneous estimation of simultaneous equations. Econometrica 30(1): 54–78.

259. Zellner, A. and E.H. Thornber, 1966. Computational accuracy and estimation of simultaneous equations econometric models. Econometrica, 34(3): 727–729.