Computers, Programming and Dynamic General Equilibrium Macroeconomic Modeling

Bongers, Anelí and Molinari, Benedetto and Torres, José L.

University of Malaga (Spain)

22 March 2022

Computers, Programming and Dynamic General Equilibrium Macroeconomic Modeling

Anelí Bongers, Benedetto Molinari and José L. Torres

Department of Economics and Economic History, University of Malaga, Spain

Abstract

Dynamic stochastic general equilibrium (DSGE) models nowadays undertake the bulk of macroeconomic analysis. Their widespread use during the last 40 years reflects their usefulness as a scientific laboratory in which to study the aggregate economy and its responses to different shocks, to carry out counterfactual experiments and to perform policy evaluation. A key characteristic of DSGE models is that their computation is numerical and requires intensive computational power and the handling of numerical methods. In fact, the main advances in macroeconomic modeling since the 1980s have been possible only because of the increasing computational power of computers, which has supported the expansion of DSGE models as more and more accurate reproductions of the actual economy, thus becoming the prevailing modeling strategy and the dominant paradigm in contemporaneous macroeconomics. Along with DSGE models, specific computer languages have been developed to facilitate simulations, estimations and comparisons of the aggregate economies represented by DSGE models. Knowledge of these languages, together with expertise in programming and computers, has become an essential part of the profession for macroeconomists at both the academic and the professional level.

Keywords: Dynamic stochastic general equilibrium models; Computers; Programming languages; Codes; Computational economics; Dynare.

JEL Classification: C61; C63; C88; E37

1. Introduction

The dynamic stochastic general equilibrium (DSGE) framework is the mainstream of contemporaneous macroeconomic analysis. Although this framework has important theoretical shortcomings and has proven not to be well-suited to explaining several key aspects of the aggregate economy, DSGE models are still used everywhere from academia to central banks and from financial and international institutions to governmental services and departments of the public administration, supporting policy choices. They are the most common tool for policy evaluations and forecasts and constitute the forefront of state-of-the-art macroeconomic modeling.

DSGE models conquered modern macroeconomics because of a number of advantages that they had with respect to the macro models in circulation before them, either the econometric models formed by big systems of reduced-form equations that central banks employed until the 1980s/1990s or the elegant but not comprehensive models of economic growth developed by academic researchers during the 1960s and the 1970s. At the time, they improved upon the existing macro models by providing a unified and stylish mathematical framework to approach almost every research question regarding the aggregate economy, thus charming macroeconomists and policy makers. They were robust to Lucas's (1976) critique because agents' decisions were micro-founded and self-interacting in the dynamic framework (see Sergi, 2018). Their underlying theoretical framework was adequate not only for describing the reaction of the aggregate economy to stochastic shocks in the short run (the aggregate dynamics) but also for characterizing changes in the (deterministic) long-run equilibrium in response to variations of exogenous variables and parameters (the steady state). Thus, DSGE models could also be used to study structural changes, for example tax changes or the introduction of a new tax. Exogenous shocks could be either stochastic or deterministic, thus delivering impulse response function analysis or transition analysis, like economic growth models. A variety of rigidities (real, nominal and informative) and frictions could be introduced, thus making the models suitable for quantitative analyses of several research questions and aggregate phenomena.

The popularity of DSGE models began when, at the beginning of the 1980s, Finn Kydland and Edward Prescott (1982a) provided the numerical computation of the real business cycle (RBC) model, a very stylized DSGE model that eventually won them the Nobel prize for their contribution to the field of macroeconomics. From that initial contribution, which represented a basic neoclassical economy with perfect competition and flexible prices, DSGE models were extended to embed all aspects and theories regarding the aggregate economy, from Keynesian characteristics like nominal rigidities in prices to real rigidities, information rigidities, deviations from rational expectations, the inclusion of side sectors (financial, exterior and R&D), all types of technological progress (investment-specific, labor-augmenting and automation), heterogeneous agents, non-Ricardian agents and so on. As a result, the complexity of the theoretical apparatus behind DSGE models increased substantially, as did the number of variables, parameters and shocks appearing in the models. It is easy to realize that the continuous increase in the

computational power of personal computers and the advances in programming languages have played a key role in paving the road to success for the DSGE framework, the models of which have become increasingly complex, computationally burdensome and challenging for econometricians over time.

The close link between computing and macroeconomic modeling is justified by the fact that DSGE models are represented by a complex framework composed of a system of highly non-linear equations with forward-looking variables that, in general, have no closed-form solution and have to be solved numerically, hence the absolute necessity of computers, programming languages and numerical methods for macroeconomic analysis. This means that the computational power of computers and the development of programming languages and most likely specific estimation/simulation software are necessary tools for the use of DSGE models in practice. This is especially true as the complexity of DSGE models has increased over time with the emergence of New Keynesian models and the introduction of features like heterogeneous agents, nominal and real rigidities and so on. All these developments of the macroeconomic theory only strengthen the importance of having computers and software to develop new DSGE models, study them and, eventually, take them to the data for their empirical validation.

It can be argued that the DSGE revolution started thanks to computation and to the pioneering code contributions by some leading economists with programming skills from some particular American universities (Carnegie-Mellon, Rochester and Minnesota, in particular). These leading economists understood the importance of incorporating the knowledge of programming languages into the new type of models not only as an empirical tool for econometric analysis but also as a tool for theoretical macroeconomic modeling. Nevertheless, the theoretical framework on which DSGE models are based is not new to economics. Indeed, the basic and canonical theoretical framework was developed by Frank P. Ramsey almost a century ago, in the late 1920s (Ramsey, 1928). However, then, its mathematical complexity and the absence of computers to obtain numerical solutions and model simulations made their usage impractical for a long time and hindered their adoption in macroeconomic modeling. Although some theoretical advances were made during the 1960s and 1970s based on the Ramsey model, and new solutions and numerical methods were developed, the barrier imposed by the lack of computational power (hardware and software) limited their application and prevented

them from spreading in the profession, although they accounted for the main theoretical framework for the optimal growth literature developed during the 1970s.

It was not until the 1980s, with the seminal work of Kydland and Prescott (1982a), that the DSGE revolution started. This first numerical application of DSGE models consisted of quantifying the relative importance of technological shocks in explaining the business cycle. However, this new approach to macroeconomic analysis was restricted to those economists with access to computers and with programming skills, a rare and limited combination at the beginning of the 1980s, although this situation changed rapidly. The computer language used in these first applications was Fortran, a compiled programming language that was well extended across a number of fields but not very common in economics at the time and was run using third-generation computers. It can be argued that the DSGE revolution in its early days was a matter for a handful of the foremost clever economists, in some leading departments, who learned compiled programming languages, such as Fortran or C, as a new basic tool for theoretical macroeconomic analysis as the only way to deal with these dynamic, micro-founded, rational expectations forward-looking agents: general equilibrium models.

DSGE modeling, the intensive use of computers and the acquisition of programming skills changed rapidly and, a few years later, by the beginning of 1990s, this approach became the dominant one in macroeconomics. During the 1990s, programming language skills and the extended use of scripting matrix-oriented programming languages, such as MATLAB, were an essential part of the toolbox for all macroeconomists, not only those pursuing empirical analysis but also those contributing at the theoretical level. A further and definitive step ahead in DSGE modeling during the first years of the new century was the development of specific software for solving this type of model, such as Dynare, gEcon, YADA and IRIS. These software packages or pre-processors are not only a collection of codes with tools for solving, simulating and estimating DSGE models but packages written specifically to solve any type of DSGE model, following in some aspects the philosophy of WinSolve. The most famous and extended DSGE modeling package is without doubt Dynare, which has eliminated the traditional computational barriers to DSGE modeling, transforming the hard-task solution and simulation procedures of this type of models into a straightforward step in macroeconomic modeling. Alternatively, advances in hardware allow access to incredibly powerful calculus

machines anywhere and by anyone. As shown by Blake (2012), even an iPhone/iPad can be used to solve and carry out experiments with DSGE models.

Macroeconomic modeling during the last 40 years has been tightly driven by the penetration of computing and programming techniques in a close symbiosis with theoretical progress. As pointed out by Kocherlakota (2009), the initial simple DSGE models (RBC type) were a by-product of the computing technology limitations existing in the 1970s, and the posterior development of more advanced models of the New Keynesian DSGE type was the result of innovations in computing in recent decades. Judd (1997) discussed the relationship between computational economics and economic theory. Computers have changed the way in which macroeconomics is developing, and they are the key to current macroeconomic modeling for a number of reasons. Economics is not an experimental science, and computer modeling is the only way to conduct experiments. Computational economics is well developed in the case of the computational general equilibrium (CGE) approach, in which standard software packages, such as GAMS and GEMPACK, have also been developed, although they are mostly static models. New specific software for DSGE modeling fills this gap, transforming macroeconomics into a computational field.

The structure of the rest of the paper is as follows. Section 2 presents a brief history of D[S]GE modeling, from the canonical model developed by Ramsey (1928) to the more recent high-scale New Keynesian models developed in the 1990s with hundreds of equations, and the strategy followed by taking the model to the data. Section 3 reviews the computing languages, codes and packages developed over time for solving dynamic macroeconomic models. Section 4 focuses on the implications of specific packages for solving the dynamic general equilibrium model for the dissemination of this type of macroeconomic modeling, mainly among graduate students. Section 5 presents some concluding remarks.

2. A brief history of D[S]GE modeling

This section presents a deliberately incomplete review of the history of DSGE modeling to illustrate how this modeling strategy has advanced over time and how this historical process has been linked to advances in computing and programming. Dynamic general equilibrium (DGE) models, either stochastic or deterministic, have become the

fundamental tool and the dominant paradigm in current macroeconomic analysis. Modern macroeconomic analysis is increasingly concerned with the construction, calibration and/or estimation and simulation of DSGE models, being the main tool for economic policy analysis. DSGE modeling starts from what we call the micro-foundations of macroeconomics, and it is, at its core, based on the rational expectations forward-looking behavior of economic agents. While the theoretical principles are not too complex to be understood by a beginner in this topic, perhaps with the exception of some mathematical techniques, solving the model and carrying out practical applications of the data are usually more difficult tasks as numerical solution methods must be used. Once the theoretical model is at hand and the equations of the model economy have been parameterized, we can proceed to its numerical solution. The usual procedure consists of calibrating the parameters of the model using previous information, matching some key ratios or moments provided by the data or, more recently, estimating the parameters using maximum likelihood or Bayesian techniques. However, the main problem posed by DSGE models is that they do not have a closed analytical solution—except for some very simple and unrealistic examples of limited interest—and a numerical solution approach is needed, which necessitates the use of computer software and an adequate level of computer skills. In the next section, we will show how these advances have been directly linked with the development of programming languages and their learning by economists and especially with the development of specific packages for macroeconomic modeling.

2.1. From DGE to DSGE

The foundation of D[S]GE models is the model developed by Frank Ramsey (1928), the so-called Ramsey optimal growth model. Although Ramsey was not an economist himself but a professor of mathematics at the University of Cambridge, he was a friend of Keynes, who introduced him to thinking about economic problems, such as optimal taxation and the optimal saving rate for an economy. He built an economic model in which consumers are infinitely-lived individuals or families who maximize their utility, adopting a micro-foundation approach to macroeconomic modeling for the very first time. However, the mathematics of the model prevented its adoption by economists at the time as Ramsey used calculus of variations and other mathematical techniques that were not part of the standard knowledge of economists. As a consequence, the seminal contribution of Ramsey remained in oblivion until 1965, when David Cass (1965) and Tjalling C.

Koopmans (1965) in parallel recovered the Ramsey model. These contributions gave rise to the so-called Ramsey–Cass–Koopmans model, which constituted the base of the optimal growth model, a widely extended theoretical framework used to study a number of economic problems.

The optimal growth literature was one of the main lines of macroeconomic research during the second half of the 1960s and the 1970s. Although no direct data application was made, this fixed dynamic optimization technique would be used in the later handling of DSGE models. In short, the DSGE model is just the standard balance growth model extended with a stochastic component, such as the one developed by Brock and Mirman (1972). The introduction of stochastic components into the optimal growth model allowed the transition from the long run to the short run. This permitted the use of the same model for studying both economic growth and business cycles. The use of optimal growth models was common during the 1970s, penetrating macroeconomic modeling and using this theoretical framework to study a variety of economic issues, from human capital (Uzawa, 1965) to natural resources (Dasgupta and Heal, 1974). The microfoundations of the optimal growth model based on Ramsey's model helped to face Lucas's (1976) critique of the traditional approach to policy evaluation. However, these models remained at a theoretical stage, with economic analysis mainly being performed through the graphical representation, the phase diagram, of a model reduced to a system of differential equations. The limitations of macroeconomic modeling are related to the progress of computation and access to computers by economists during the 1970s, although the dawn of third-generation computers (a scaled-down version of mainframe computers) paved the road to the numerical solution of the model (the appearance of the IBM PC took place in 1981).

The connection between the stochastic growth model and computing involves the difficulties in solving this type of models. These difficulties arise mainly from the existence of non-linearities provoked by the technology (production function) or the household utility function. These non-linearities disappear only in the case in which utility is logarithmic and the capital depreciates fully in a single period. In this case, the model becomes log-linear and a closed-form solution can be obtained, but this is not the case for more general specifications. Those features required a radical change in the way in which macroeconomic analysis had to be carried out, incorporating computers into the traditional paper and pencil tools as an indispensable and essential new instrument. As

Lucas (1980) stated, the task of macroeconomics should be "… to write a FORTRAN program that will accept specific economic policy rules as input and will generate as output statistics describing the operating characteristics of time series which are predicted to results from these policies."

Different numerical methods were used to deal with the non-linearities of the model. Following the seminal work by Kydland and Prescott (1982a), who used linear–quadratic (LQ) approximation, a number of other important contributions appeared in the 1980s, including those of Long and Plosser (1983), extending the model to economic sectors, Hansen (1985), incorporating indivisible labor, and Christiano (1988) and Altug (1989), providing the first estimations of a DSGE model using maximum likelihood techniques. Kydland and Prescott (1982a) used linear–quadratic approximation to the model around the steady state. King, Plosser and Rebelo (1987) and Christiano (1988) used log-linear–quadratic approximation. See Taylor and Uhlig (1990) for a review of the existing alternative methods. Importantly, most of these solution approaches are numerical methods, given the low cost of computation.

Another important contribution was the development of the first solution method for a rational expectations linear model by Blanchard and Kahn (1980). Later, alternative solution methods were developed, such as those by Uhlig (1999), Klein (2000) and Sims (2001). Blanchard and Kahn's (1980) solution method established a straightforward and standard procedure for solving relatively simple DSGE models and was an additional determinant contributing to the spread of the RBC model across the profession. The toolbox developed by Uhlig (1999), published in 1995, gave another important impulse to the spread of DSGE modeling by providing simple rules for the log-linearization of non-linear models and a new solution method.

2.2. From RBC to New Keynesian models

The standard RBC is a relatively simple model, with no government, no money, no frictions or adjustment costs and only a few endogenous variables (eight variables for a decentralized economy and six for a centralized economy, including the aggregate productivity stochastic process); thus, it involves only a few equations. Indeed, most of models solved with numerical methods during the 1980s adopted the neoclassical framework of perfect competition and flexible prices as a well-defined benchmark to

explain the regularities of business cycles. This approach was not chosen willingly by researchers. It was rather a constraint on the economic theory imposed by the existing computers' computational power and the availability of programming tools to solve dynamic models. As pointed out by Kocherlakota (2009), the complexity of theoretical models in the 1980s was constrained by the computing technologies. Given the computing technologies existing at that time, only simple models could be solved numerically. More complex models were hard to solve as they were time consuming and required complex numerical and computational techniques. As a matter of fact, Kocherlakota (2009) argued that the whole freshwater–saltwater division of the 1980s was a consequence of the limitations in computing technologies and numerical techniques at the time, and that division was eliminated in subsequent years thanks to better computers. This had deep implications for policy recommendations as the result of supporting policies of no optimal economies was not a theoretical principle but a result of the level of complexity of the macroeconomic models that could be solved with the existing computational techniques. The solution of more elaborate DSGE models is very computation demanding and, hence, the complexity of the theoretical framework is restricted by the computing power. Kocherlakota (2009) was convinced that the progress of computers and programming was the key ingredient for the development of more advanced and, in some cases, realistic and better theoretical models. The incorporation of nominal and real rigidities, money, borrowing restrictions, imperfect markets, financial markets and so on implies an increase in the complexity of the model and hence the demand for better and more time-efficient computing techniques and faster computers. Additionally, the advances in computing not only allowed more complex theoretical models but also introduced important innovations into the way in which models are taken to the data.

All these elements gave rise to the birth of the so-called New Keynesian DSGE models, of which imperfect competition, money, nominal and real rigidities, adjustment costs and so on are the fundamentals pillars. New Keynesian DSGE models were initially developed by Rotemberg (1982), Mankiw (1985), Svensson (1986) and Blanchard and Kiyotaki (1987). However, these models were much bigger and more difficult to solve numerically than the canonical neoclassical RBC-type model. However, progress in computing, the introduction of new numerical methods and the availability of a significant and growing set of codes contributed to making the solution of more complex

models accessible, transforming New Keynesian (NK) DSGE models into the standard tool for macroeconomic policy analysis. NK-DSGE models are much bigger than neoclassical DSGE models because they include more endogenous variables (all monetary variables) and therefore larger systems of equations and a number of adjustment processes that also contributed to a significant increase in the number of parameters. The increasing number of parameters also called for new methods for estimating DSGE models.

2.3. From calibration to estimation

DSGE models do not have closed-form explicit solutions, except for some simple cases (logarithmic utility functions and full depreciation of capital). This simply means that DSGE models cannot be solved directly by hand with paper and pencil techniques. Instead, DSGE modeling requires the use of numerical and computational methods to obtain approximate solutions for simulating the variables of the model. DSGE models have two key characteristics: a non-linear system of dynamic equations and expectations about future endogenous variables. The key to solving a DSGE model consists of representing functional forms for the control variables (for instance, consumption) as a function of lagged state variables (for instance, capital stock). Once we have these functions, the system becomes recursive; then, given the initial values for the state variables, the dynamic process for the control variables can be generated. This are the so-called decision rules or policy functions. The terms decision rule and policy function refer to functional equations, that is, functions of functions, describing the dynamics of the forward-looking control variables.

To obtain a numerical solution to DSGE models, first, it is necessary to assign values to the parameters of the models. In general, the equations of a model have three components: unknowns (endogenous variables, the value of which we are looking for), exogenous variables (which are assumed to be a fixed number or to follow a stochastic process) and parameters. The parameters of the model are assumed to be "deep" parameters, that is, constants. To be precise, it is assumed that the values of the parameters do not depend on the sample period as they are deep parameters (invariant to policy changes and therefore not affected by Lucas's critique). Two methods for assigning values to parameters have been used in the literature: calibration and estimation (or a combination of the two).

Initially, the strategy was so-called calibration, a direct and relatively easy procedure to give a value to the parameters of the model. Indeed, this "estimation" technique was another of the innovations of the paper by Kydland and Prescott (1982a). Calibration can be defined as a method to assign values to the deep parameters of a DSGE model using all the a priori information. More precisely, calibration is an econometric technique to take models to the data. DSGE models typically have identification problems that complicate the estimation of all the parameters; thus, it is natural that the first approach was based on calibration as the estimation of the parameters could lead to inconsistent and unreasonable parameter values.

In the canonical neoclassical or RBC model, we typically have four types of parameters: technological parameters, preference parameters, steady-state parameters and auxiliary or nuisance parameters related to the stochastic process for aggregate productivity. As the number of equations is small, the number of parameters is also reduced and can easily be estimated (calibrated) using national accounts or key ratios. However, a New Keynesian DSGE model has a large number of additional parameters: adjustment cost and price parameters, heterogeneous agents' share, monetary and fiscal policy parameters, mark-ups and so on. These make the calibration method less appealing and robust, and more sophisticated estimation methods are required.

Calibration can be performed using different alternative approaches. First, we can use the values of other works that have calibrated and/or estimated a similar DSGE model for the same economy. This is a very frequent approach in academic papers. The problems of this approach relate to the use of different models, different economies and so on. Second, we can use national accounts for some key parameters, for instance the technological parameters of a Cobb–Douglas production function. This parameter has an interpretation in terms of the proportion of capital income in the total rent. These data can be found in national accounts and are typically used for the calibration of the technological parameter representing the capital–output elasticity. Third, we can use estimation from econometric (both micro and macro) studies. However, this approach also has drawbacks as the estimated parameter values depend on the econometric technique, the variables, the sample period and so on. Additionally, these estimations involve independent equations, not integrated ones as used in the general equilibrium model. Finally, fourth, we can use equilibrium conditions from the model or steady-state relationships (the model-based or internal calibration approach). Using the equations of the model and a target for a variable

or a key ratio, it is possible to obtain the corresponding values for the parameters in those equations. In this case, some parameters are considered as variables and their value comes from solving a static version of the model, in which some endogenous variables are changed by the data.

However, some authors moved ahead rapidly from calibration to econometric estimation and used maximum likelihood methods for estimating the parameters (or some of them) of the model. Christiano (1988), Altug (1989), Bencivenga (1992), McGrattan (1994), Ireland (1997, 2001a, b, 2004), McGrattan, Rogerson and Wright (1997), DeJong et al. (2000) and Kim (2000) provided examples of structural parameter estimation of DSGE models using maximum likelihood methods. However, with the development of New Keynesian DSGE models, estimation of the parameters has been the standard approach. Today, DSGE models are estimated mainly using Bayesian techniques (Rabanal and Rubio-Ramirez, 2003; Smets and Wouters, 2003). The popularity and spread of Bayesian estimation of DSGE models was due to the appearance of Dynare, which allows the estimation of DSGE models using ML and Bayesian Markov change Monte Carlo (MCMC) estimation using the Metropolis–Hastings algorithm.

With the increasing complexity of DSGE models and the rising number of parameters for which little a priori information is available, the Bayesian approach is becoming increasingly popular for DSGE model estimation. Bayesian estimation can be considered to be somewhere between calibration and maximum likelihood estimation. In fact, calibration is just the specification of a prior (the first step in the Bayesian approach). Conversely, the Bayesian approach confronts the model with the data, as ML does. The Bayesian approach allows us to interfere in the estimation process. In fact, priors can be interpreted as weights in the likelihood function, giving greater importance to certain areas of the parameter subspace. One of the advantages of the Bayesian approach over the ML method is that it avoids peaking at strange points where the likelihood peaks; that is, it avoids situations with absurd parameter estimates. Rabanal and Rubio-Ramirez (2003) and Smets and Wouters (2003) were the pioneers in estimating a DSGE model with novel Bayesian techniques for economists. Fernández-Villaverde (2010) provided an excellent review of these estimation techniques.

2.4. Programming and the teaching of dynamic macroeconomics

Finally, computing and programming advances have allowed the development of simple tools for DSGE modeling that can be used not only by graduate but also by undergraduate students; therefore, they have implications not only for academic research but also for the macroeconomic theory taught at the undergraduate level. The work of Hobbs and Judge (1992) is a good starting point for the discussion about the contribution of computer-assisted learning to the teaching of economics. Computers are used widely in quantitative methods and econometrics but not for the teaching of economics. This is crucial for advanced macroeconomics, in which there is a real need for more accessible approaches to teaching DSGE models to undergraduates. Indeed, there is an important current debate about teaching DSGE models at universities. Solis-García (2018) defended the teaching of DSGE models at the undergraduate level. Neumuller, Rothschild and Weerapana (2018) recognized the existing gap between undergraduate macroeconomics and graduate macroeconomics (a gap that does not exist for microeconomics or econometrics). This is an important issue as macroeconomics teaching in current graduate programs builds on DSGE models, whereas typical undergraduate macroeconomics courses are based on the traditional IS-LM framework. The difficulties in teaching DSGE models to undergraduate students arise from: i) the language of general equilibrium concepts; ii) the mathematical background; and iii) computer language skills. Whereas the first two elements are obstacles that are relatively easily overcome, the third constitutes the most important barrier to the teaching of DSGE models in the most advanced undergraduate macroeconomics courses as a numerical solution is needed; hence, students are forced to solve the model computationally, for which they must use certain software. The standard software used for solving DSGE models is MATLAB and, more recently, Python and Julia, which necessitate a level of computer skills that is not generally possessed by undergraduate students. Some advances have been made, although these are very limited. For instance, Chu (2018) proposed a method that serves as a bridge between the Solow model and the Romer model, in which the mathematical derivations involve only basic calculus and algebra, and the basic principles of the Romer model are more accessible to undergraduate students in economics. Bongers, Gómez and Torres (2020) presented two alternative and easy methods for solving DSGE models in a spreadsheet such as Excel: i) the models can be solved using Excel's Solver tool, which employs a linear programming algorithm for solving a system of equations; and ii) the model can be linearized and

numerically simulated directly in a spreadsheet using an eigenvalue method, without the need to use a computer optimization algorithm.

Another interesting resource is the website created by Brian C. Jenkins (2022), on which some dynamic macroeconomic models can be solved and simulated. For each model, the parameters can be calibrated by the user, and simulated variables and impulse responses can be plotted. Finally, Bongers, Gómez and Torres (2021) developed a web page on which several dynamic macroeconomic models can be solved in Excel, including the standard RBC model, the investment-specific technological change model and the Ramsey optimal growth model.

3. Codes, models and computers

As stated above, the macroeconomic theoretical and empirical developments during the last four decades have largely been driven by advances in programming languages and specific software and hardware. The reason for that relationship can be found in the characteristics of the new dominant macroeconomic paradigm, based on the forward-looking rational expectations dynamic general equilibrium model, which can only be solved using numerical methods and computing techniques. However, the connection between economics and computation is not new. Indeed, attempts to use some types of computing machine in economics occurred prior to the invention of analogic computers. Taylor and Uhlig (1990) recognized that it is the fact that computing power has become faster and cheaper that enables macroeconomists to study more complex models and apply them for policy analysis. Initially, it was very difficult to solve a DSGE model. Today, it is a much easier task thanks to the development of many techniques and to the existence of specific computer programs developed by clever people. Parallel to the development of programming, a large variety of alternative numerical methods have been incorporated into macroeconomic analysis: local methods (i.e. the perturbation method) versus global methods (i.e. projection methods: dynamic programming, the Chebyshev polynomial method, the finite-elements method, the extended path method, parameterized expectations, neural networks, etc.).

Nowadays, economists have a wide variety of computer software codes in different languages and some DSGE-specific packages. We classify these sources into three types: codes in compiled languages, codes in scripting (interpreted) languages and packages (or

more exactly pre-processors). Macroeconomic modeling has been heavily based on codes in compiled and scripting languages, made available by their developers, and only recently have packages have been developed, Dynare being the most popular. This is in contrast to, for instance, econometrics, in which packages, both open access and commercial, dominate.

Kendrick and Amman (1999) classified computer programming languages into three groups: a) high-level languages (GAUSS, MATLAB, Maple, Mathematica and GAMS); b) low-level languages (Basic, Fortran, C/C++ and Java); and c) languages for programming graphical user interfaces (GUI), such as Java, VisualBasic and Visual C++. Each type of programming language has pros and cons. Flexibility, runtime speed, ease of learning and so on are all factors to consider. Codes in compiled languages, such as Fortran or C/C++, are very flexible and can be adapted to the particular problem at hand. These codes can be modified by users, increasing the number of available codes. Alternatively, codes in compiled languages are freely available, and, more recently, some advanced scripting languages have emerged that are free and offer a number of advantages over the previous commercial scripting languages that had been widely used in macroeconomic modeling. Finally, pre-processors or specific packages have significantly increased the availability of programming tools for DSGE modeling.

3.1. Computational economics in the pre-digital era

Computation was an extremely difficult task before the advent of digital machines, and reliable analog computers were scarce. One type of analog computer intended for economic applications was hydraulic computers. The first idea about using hydraulic machines for economic modeling came from Irving Fisher. In his PhD thesis (Yale University, 1891), Fisher presented a hydraulic apparatus for computing equilibrium prices and the resulting distribution of society's endowments among economic agents (Brainard and Scarf, 2005).

The first attempt at computing macroeconomic models was performed by A. W. H. Phillips in 1949 (Phillips, 1950). The Phillips machine, also known as the Monetary National Income Analogue Computer (MONIAC), is a mechanical device built with tubes, pipes, valves, pumps, tanks, electrodes and servo-mechanisms. In the construction of the original machine, Phillips used different materials from Lancaster bombers: the

tubes and tanks were made of Perspex from aircraft windscreens, and the pump was the landing gear pump. Colored water flowed in the tanks and tubes. The machine represents an economy ruled by an aggregate model formed of eleven equations, based on the canonical Mundell–Fleming model. There is also the possibility of connecting two mirror machines to manage a two-country model. The MONIAC was the first analog computer to solve the non-linear differential equations of the IS-LM model (Bollard, 2011).

During the 1950s, other attempts to solve dynamic models numerically were undertaken based on the development of electro-analog computers. The idea consisted of the possibilities of applying electrical analog computing techniques to a broad number of economic issues, that is, building an electrical circuit for an electro-analog simulation of dynamic economic models. Morehouse, Strotz and Horwitz (1950) developed an electrical circuit for the simulation of an inventory model. Enke (1951) used this electro-analog approach for simulating spatially separated markets, whereas Strotz, Calvert and Morehouse (1951) simulated a stochastic national income model, and Strotz, McAnulty and Naines (1953) developed an electro-analog solution for the non-linear business cycle model proposed by Goodwin (1951). See Raybaut (2020) for a review of this electric-analog computing simulation approach.

Apart from the MONIAC and electrical circuits, little use of computational techniques for solving dynamic macroeconomic models occurred in the 1960s and 1970s. This contrasts with applied economics, in which computers were used extensively in those decades for estimating a variety of econometric models. This was natural as econometric analysis and statistical applications to economics have a long tradition. Nevertheless, computers for economic simulations were used during the 1950s, such as the development of system dynamics by Forrester (1958) using an IBM 704 computer or by Adelman and Adelman (1959) using an IMB 650 computer. During the 1960s, the use of mainframe computers extended among economists as they became more accessible, although they were mainly used for empirical analysis. Finally, another important change in computational economics occurred during the 1970s with the development of specific software, that is, the General Algebraic Modeling System (GAMS) for computable general equilibrium (CGE) models.

3.2. The 1980s

The incorporation of computing techniques into theoretical dynamic macroeconomic modeling started at the end of the 1970s. In the first applications, the computer language code used in dynamic macroeconomics was Fortran. The penetration of Fortran in economics is explained by its popularity among numerical mathematicians and engineers. Some economists started, for the first time, to write Fortran codes for solving and simulating a dynamic general equilibrium macroeconomic model. The standard numerical approach used was linear–quadratic (LQ) approximation. The initial macroeconomic modeling programming in Fortran expanded rapidly among several economists, although it was concentrated in a few universities. The initial contribution by Kydland and Prescott (1982a) was soon followed by those by John B. Long, Charles I. Plosser, Gary D. Hansen, Lawrence J. Christiano, Ellen R. McGrattan and Sumru G. Altug, among others, expanding the solution techniques and the corresponding Fortran code. Kydland and Prescott (1982b) develop a DOS executable program (the operating system in personal computers at the time) to perform the calculations in their paper and to generate impulse–response functions, in which some parameters can be entered interactively; they also created a web interface that is no longer active.

At the beginning of the 1980s, computers were fourth-generation machines (based on microprocessors). However, the introduction of the IMB PC, a fifth-generation computer, in 1981 changed the scene completely, with an easily accessible, extremely reliable and powerful computer, albeit with some data storage limitations. These new-generation computers supported all the existing programming languages, such as C and Fortran, as well as the development of new code and packages in PC DOS. The introduction of hard disks in 1983 solved the problems with data storage. As a consequence, the transition from mainframes to PCs was very fast, increasing the access to computers.

The combination of all these elements radically transformed macroeconomics in only a decade, and less than 10 years after the seminal contribution by Kydland and Prescott (1982a), the RBC and the DSGE approach in general spread widely across a large number of departments and researchers around the world, emerging as the dominant paradigm in modern macroeconomics.

3.3. The 1990s

During the 1990s, the use of DSGE models expanded rapidly among the academic community due to the enhanced access to computers, the development of specific software and the availability of programming codes from individual contributors first and then from more ambitious teams that developed specific packages for solving DSGE models without the need for computer programming skills. Developments in computer languages were also rapidly incorporated into macroeconomic modeling. The variety of programming languages used by economists expanded, not only to compiled languages but also to some new scripting languages, such as MATLAB, R, Mathematica and Octave, that were rapidly incorporated into economics. Initial codes in Fortran were translated into GAUSS, OX, MATLAB, R and, more recently, Python and Julia. Codes in other languages, such as TROLL and RATS, were also developed but in a more limited quantity. The most widely used scripting language adopted by macroeconomists has been MATLAB, leading with some initial key free code developed by leading practitioners (see Nerlove, 2004).

Nowadays, all macroeconomists have skills in one or a number of computer languages as an essential basic tool of their profession as an economic theoretical background or as econometric techniques. However, some barriers to DSGE modeling remained in the 1990s as it required computing skills that were substantial and no specific software was available. This contrasted with the extended use of computers in statistics and econometrics, with the availability of specific software, such as SAS, SPSS and TROLL from the early 1970s and later Stata and other econometrics-specific packages. A remarkable attempt to fill this gap at the beginning of 1990s was the development of *WinSolve* for solving and simulating non-linear models. This was the first specific package for DSGE and other dynamic macroeconomic modeling, but it was commercial software with very limited success.

The spread of the user community started with some leading contributors offering publicly available code for solving DSGE models: Ellen McGrattan, Harald Uhlig, Christopher Sims, Frank Schorfheide and Christian Zimmermann, just to cite a few among many. Whereas the beginnings were difficult due to limited software, the publicly available contributions developed in MATLAB, R, GAUSS, C and Fortran helped an increasing number of economists to cross those barriers. In parallel, a number of websites collected a variety of sources, including codes for different solution techniques for

solving DSGE models. Another remarkable contribution was the website developed by Christian Zimmermann named Quantitative Macroeconomics and Real Business Cycle (QM&RBC), which collected a number of contributions and codes. DSGE-NET is an international research network for DSGE modeling and monetary and fiscal policy. It includes a number of programs in different computer languages by a number of contributors. The community is undergoing continuous expansion. More recent contributions are the website QuantEcon developed by J. Perla, T. Sargent and J. Stachurski for quantitative economic modeling and the tutorial in Julia by Bradley Setzler.

The first package created specifically to solve dynamic macroeconomic models was SoWhat by Stefan Bachmann and Holger Strulik (1992). It is a computer package for solving dynamic models numerically with a simple menu for simulating dynamic models. The use of SoWhat is very intuitive as no programming skills are needed. The user simply needs to introduce the equations of the models, the initial values for the endogenous variables and the values for the parameters. Endogenous variables are automatically detected by the software. The program uses the Runge–Kutta algorithm for simulating the dynamic system and includes some options regarding the simulating errors and the speed of the simulation. This package can be used for simulating models such as Solow's growth model but not more sophisticated models with rational optimizer agents.

Finally, another important contribution during the 1990s was Harald Uhlig's toolkit (1999), the first version of which was published in 1995. This is a collection of MATLAB codes to solve for the recursive equilibrium law of motion with the method of undermined coefficients for non-linear dynamic stochastic models. This toolkit has become very popular among DSGE practitioners and an invaluable source for PhD students, especially for model log-linearization and solution techniques.

3.4. The 21st century

The new century started with the preponderance of MATLAB as the main scripting language for developing code for dynamic macroeconomic modeling, although Fortran and GAUSS were still used. This is, for instance, the case of the contributions by Paul Klein with the solab (solving difference equations) codes developed in MATLAB, Fortran and GAUSS. New researchers chose MATLAB as a computing language mainly

due to the large quantity of available free code and other resources for numerical methods previously developed by a number of authors. One of the main advantages of MATLAB is that it is a natural environment notation for writing linear algebra. However, it is a commercial language (although the alternative option of Octave exists), poor at dealing with data analysis and inferior to incoming new scripting language ecosystems that have experimented with rapid and continuous expansion in several fields, including economics.

Indeed, two new languages have been developed in recent years, and macroeconomists have incorporated these new computing tools into the toolbox for macroeconomic modeling. These new computing languages are Python and Julia. Python is an object-oriented programming language, adopted by some economists as an alternative to MATLAB. Julia is a more promising programming language in economics given its similarities to MATLAB and its faster computation speed compared with Python. One of the main supporters, Thomas J. Sargent, is the founder of QuantEcon, a platform that advances pedagogy in quantitative economics using both Julia and Python, including open-source code for economic modeling. Sargent (2016) considered that the next generation of macroeconomic models will be very computationally intensive, with large datasets and many variables. These macroeconomic models and their forecasts help to solve large constrained optimization problems using massive datasets to inform policy analysis; hence, the availability of programming languages such as Julia will determine further advances in macroeconomic models.

3.4.1. Python

Python is a general-purpose interpreted programming language with an object-oriented approach that was initially developed by Guido van Rossum in 1991. A fully revised version 2.0 was released in 2000. In the last years, Python has become one of the most popular programming languages. It is suitable for a variety of applications because it is supported by a vast collection of scientific libraries that are continuously updated by its community. During the last five to seven years, Python has also been incorporated into the computing toolbox in economics. As a matter of fact, the dynamic programming language Python is well suited to use in economics and, in particular, econometrics, which typically involves matrix-based calculations. The scientific packages for numerical programming (NumPy), data preparation (pandas) and symbolic programming (Sympy)

with their many well-known extensions provide an ideal framework for working scientifically in the field of economics.

Two remarkable sources of Python libraries designed to solve and compute DSGE models are the Python Macroeconomics Laboratory (PyMacLab) and QuantEcon.py. The latter is an open-source Python code library for economics and finance that is financially supported by the Alfred P. Sloan Foundation and promoted by the Nobel prizewinner Thomas J. Sargent. The library QuantEcon.py, together with QuantEcon.jl (using the same codes but in the Julia language), form QuantEcon (Quantitative Economics), a fiscally supported project dedicated to the development and documentation of modern open-source computational tools for economics, econometrics and decision making. Another contributor is Brian C. Jenkins, who developed a website on which DSGE models can be simulated. On his website, users can choose the parameter values for the simulation and other simulation-specific settings, and simulation results can be visualized or downloaded, but there is no flexibility regarding the type of models to be solved.

As an example, to compute DSGE models, Dynare still outweighs Python by a factor of 12, according to a Google Scholar search with the terms Python "AND" DSGE (359 results) vs. Dynare "AND" DSGE (4960 results). Nonetheless, the growing success of Python as a programming language suggests that this situation may quickly change, although Julia is a formidable competitor. As argued by the PyMacLab team, there are several clear-cut advantages in developing and using software written in Python rather than any other programming language. First, Python is rapidly turning into the language with the best supply of ready-to-use libraries. Second, it glues well to traditional scientific languages, thus allowing existing source codes in other languages (e.g. Fortran and C++) to be called inside Python scripts as if they were normal Python routines. In addition, Python code can easily be embedded into the programming codes of html web pages, Flash applications or smartphone/tablet apps (both HTML5 and Java), which are all compatible with Python. Third, differently from Java and G++, Python is interpreted and not compiled, thus making the programming experience much more seamless, interactive and transparent. This turns Python into a so-called rapid application development (RAD) tool. Python also has limitations. For example, the programming languages of Python 2 and Python 3 are incompatible. Additionally, Python is slower than Java and C+, even though it is faster than MATLAB and GNU-Octave (see Aldrich et al., 2011). Finally,

Python is not self-contained but requires module support that sometimes lacks adequate inputs.

3.4.2. Julia

A more recent programming language alternative with a number of advantages over Python is Julia. Julia is a high-level, high-performance, free and open-source dynamic programming language for technical computing, with syntax that is familiar to users of other technical computing environments. In particular, its syntax is quite similar to that of MATLAB, which is a significant advantage. The similarity of the syntax means that a lot of MATLAB code will run in Julia with few changes. Indeed, Julia can be defined as a combination of MATLAB and Python languages at the syntax level. However, it is a more advanced programming language given that it has just-in-time compilation characteristics and hence a computation speed close to compiled languages such as Fortran or C/C++.

Julia is a scientific computing language, and an increasing number of economists are adopting this programming language for computation-intensive tasks (e.g., Tom Sargent and the NY FRB). It is a close substitute for MATLAB, and the cost of switching from MATLAB to Julia is somewhat modest since Julia's syntax is quite similar to MATLAB syntax after changing array references from parentheses to square brackets (e.g., "A(2, 2)" in MATLAB is "A[2, 2]" in Julia and most other languages), though there are important differences. Julia also competes with Python, R and C++, among other languages, as a fast-computational tool. Julia's advantages are that it is modern, elegant, open source and much faster than MATLAB. Its disadvantage is that it is a young language, so its syntax is evolving, its user community is smaller and some features are still undergoing development.

Julia has many advantages over other scripting languages and for this reason is extensively used in industries and in research with a fast penetration speed. Recently, the Federal Reserve of New York open sourced its macroeconomic model (used for producing forecasts about key variables and conducting policy experiments). The code is written in Julia. One result of this research is DSGE.jl in GitHub, a Julia language package that facilitates the solution and Bayesian estimation of DSGE models.

An excellent tutorial for programming in Julia for economists is the manual written by Jesse Perla, Thomas J. Sargent and John Stachurski (Sargent and Stachurski, 2015; Perla, Sargent and Stachurski, 2022). This tutorial can be found on the website https://julia.quantecon.org/intro.html. As well as providing an excellent introduction to the language, this is also a complete macroeconomic textbook with concepts illustrated in Julia. Another source of codes for solving DSGE models in Julia can be found in the book by Caraiani (2019). Furthermore, Bradley Setzler has elaborated some Julia economics tutorials on structural econometrics (https://juliaeconomics.com/tutorials/).

The high running speed of Julia compared with MATLAB is a key aspect for new developments in macroeconomic modeling using this new programming language. For example, Thomas Hasenzagl, Filippo Pellegrino, Lucrezia Reichlin and Giovanni Ricco switch from MATLAB to Julia for computing models using real-time data for monitoring the economy. They chose Julia due to the computational intensity of the problem at hand as Julia significantly improves the computational efficiency and speed of the nowcasting model. This framework employs a number of different algorithms, including an expectation maximization (EM) algorithm for dynamic factor models (DFM), a Kalman filter and smoother, and several routines used to measure the impact of each individual economic release. They also pointed out that Julia is crucial when a large number of simulations is required, such as in Bayesian estimation methods, given its reduced running time compared with MATLAB.

Table 1 summarizes the three types of computing software that are most extensively used to solve, simulate and estimate DSGE models, classified by software type and year. During the 1980s, two main languages were used: the Fortran compiled language and the GAUSS scripting language, although the latter had little penetration in macroeconomics. The spread of DSGE modeling came about in the 1990s, with application in C/C++ and the advent of a number of scripting languages, such as MATLAB, R, Mathematica and Octave, MATLAB being the most popular among macroeconomists. During this decade, some packages for solving simple dynamic macroeconomic models, such as WinSolve and SoWhat, also appeared.

Table 1: Computing languages for DSGE modeling

	Compiled languages	Scripting languages	Packages/preprocessors
1980s	Fortran	GAUSS	
1990s	C/C++	MATLAB	WinSolve
		R	SoWhat
		Octave	
		Mathematica	
2000s		Python	Dynare
			gEcon
			IRIS
			YADA
			GEMLLIB
2010s		Julia	Dolo/Jolo

Auroba and Fernández-Villaverde (2015) used the stochastic neoclassical growth model and solved this model with the same algorithm (the value function iteration with a grid search for future optimal capital) in different languages: C, Fortran, MATLAB, Mathematica, R, Java, Python and Julia. They found that compiled languages (C and Fortran) are faster, with C being slightly faster than Fortran but with Julia (a scripting language) almost being faster than compiled languages. By contrast, Python, MATLAB and R are extremely slow. They reported that MATLAB is around 10 times slower than C++. As is usual in this type of computing exercises, the comparison was based on the execution of a specific problem and measured the solving time. However, more factors influence which programming language is the most appropriate, which depends on a number of factors: personal skills in programming languages, the language/software that is used most in the profession and the number of code sources that are publicly available for each language.

4. Packages and pre-processors: The democratization of DSGE modeling

Packages, either free or commercial, have been common in statistics and econometrics for a long time. Indeed, we can find a large number of software programs covering all the existing statistical and econometric techniques used for empirical economics. Moreover,

new econometric techniques developed by researchers have quickly been incorporated into these packages, eliminating the need to develop individual specific codes for the use of these new techniques and the necessity of knowledge of programming languages. The circumstances have been different for macroeconomic analysis. Econometricians have unified their language, and the computation for empirical estimations of models has been highly standardized. The road of macroeconomics has been very different from that of econometrics. After the initial attempts, such as the release of the SoWhat package (Bachmann and Strulik, 1992; Strulik, 1992) and Winsolve (Pierse, 1998, 2000), advances in macroeconomic computation have generally been piecemeal, with the individual development of codes to solve particular problems. This difference is explained by two factors. First, the number of numerical techniques and solution methods for working with DSGE models is large (although they produce different results, as shown by Taylor and Uhlig, 1990), and different authors have different preferences for using a particular numerical method and solution techniques. Second, theoretical models are customized, and they can have particular characteristics that require specific codes. However, the situation has changed recently, and, during the last two decades, a number of specific packages, apart from the commercial WinSolve, for solving, simulating and estimating DSGE models have been developed by different teams in a free and non-commercial fashion.

The new century has been characterized by the development of specific packages for DSGE modeling written in both compiled and scripting languages. A keystone that contributed decisively to the spread of DSGE models as the cornerstone of modern macroeconomic analysis was the development of Dynare by a CEPREMAP team. Dynare (**Dyna**mic **R**ational **E**xpectations) was developed initially in GAUSS by Michael Julliard and later extended to MATLAB, Octave and C++. More recently, the Dynare team has developed this pre-processor for DSGE modeling in Julia (Dynare.jl). It can be argued that Dynare revolutionized the profession as people with a low level of programming skills can solve and simulate DSGE models using this specific user-friendly user interface (UI) combined with MATLAB. The development of specific software such as Dynare has "democratized" the use of DSGE models as economists with few skills in computer languages but enough knowledge about the theoretical model can use these models for macroeconomic analysis, something that was restricted to a relative small "DSGE club" before the dawn of Dynare.

Apart from Dynare, other packages have been developed to handle DSGE models: gEcon, IRIS, YADA and Dolo/Jolo. gEcon is a tool for solving DGE models developed in R at the Department for Strategic Analyses at the Chancellery of the Prime Minister of the Republic of Poland, by Grzegorz Klima, Karol Podemski and Kaja Retkiewicz-Wijtiwiak. Another specific piece of software is IRIS, consisting of a toolbox for MATLAB developed by the IRIS Solutions Team since 2001, headed by Jaromir Benes for macroeconomic modeling and forecasting. YADA is another package for MATLAB developed through research at the European Central Bank since 2006. Recent packages for Python and Julia are Doco/Jolo. Finally, another contribution is GEMLLIB, developed by Pawel Kowal (2005). This is a collection of routines for C++ in GEML language to be run in MATLAB, devoted to specifying large-scale DSGE models easily, including algorithms that perform all the required symbolic computation to solve the model. GEMLLIB uses the perturbation method for solving DSGE models. However, the success of Dynare has not been reached by any other software for DSGE modeling.

4.1. WinSolve

WinSolve is a program for solving and simulating non-linear models. It is a commercial package developed by Richard G. Pierse in 2004. Whereas commercial packages have been common in statistics and econometrics, this is an exception in the case of dynamic macroeconomics or structural econometrics, for which codes have traditionally been free. This program handles a wide class of models, including both DSGE models and structural econometric models. WinSolve is a stand-alone application for the Windows operating system. The size of models that can be built is unlimited as there is no restriction on the number of variables or parameters. This software can linearize a non-linear model and use the perturbation methods to solve it. Although version 5.0 was announced in October 2016, only version 4.0, from November 2007, is available. WinSolve uses a large variety of solution algorithms and alternative numerical methods. Currently, the distribution of WinSolve by a commercial company has been announced but still not commenced.

4.2. Dynare

Without any doubt, the most popular software with the greatest impact on the development of dynamic macroeconomics is Dynare. Dynare is a free and open-source pre-processor, that is, a pre-compiler consisting of a program that processes input data (a text file) to produce output (i.e., a MATLAB program) that is used as input for another program (i.e., MATLAB). Part of Dynare is programmed in C++, and part in is programmed in MATLAB/Octave. This pre-processor uses a very simple language that allows the conversion of a DSGE model into a simple code that can be run in various programming languages, such as MATLAB or Octave, and it is expected in the near future to be run in Julia. Additionally, Dynare++ is a stand-alone program. Dynare has been developed at CEPREMAP (Centre pour la REcherche ÉconoMique et ses APplications) by a team directed by Michel Julliard, Stéphane Adjemian and Sébastian Villemot. Dynare can solve both DSGE and perfect-foresight DGE models and models with alternative expectation mechanisms, and it estimates DSGE models using both maximum likelihood and Bayesian techniques.

The success of Dynare can be attributed to a number of reasons. First, the source syntax is very friendly and simple, keeping the command instructions to a minimum. The input file for the pre-processor is a text file with a set of simple instructions and blocks. It is only necessary to provide a text "mod" file with a set of endogenous variables, a set of exogenous variables, the parameters, the values of the parameters and the equilibrium equations of the model. The set of equations is written in a similar fashion to a sheet, the only change being the way in which the time indexes are provided. For Dynare, the model is a set of equations defining the technology, the feasibility condition and the optimal decisions by economic agents (first-order conditions of maximization problems previously solved by hand). This implies that few programming skills are required and that the structure and scale of the model can be changed easily.

Second, it is based on MATLAB/Octave, scripting languages that are already used by DSGE modelers. Although the first version was developed in GAUSS, it was subsequently developed in MATLAB, a programming language used by most macroeconomists. This facilitated the adoption of Dynare given its flexibility and possibilities. Additionally, Dynare permitted researchers to concentrate on the theoretical model, without the need to integrating existing MATLAB code or create their own code, saving time and avoiding errors. On the other hand, new models, especially NK-DSGE

estimated models, have been written in Dynare, a publicly available code, to replicate the results published in articles and to explore further those collections of models with alternative specifications and values of the parameters.

Finally, Dynare is a simple yet powerful tool and is easy to learn, requiring only basic knowledge of MATLAB/Octave, so it can be used by graduate students with little knowledge of programming languages. This has contributed to the expansion of Dynare's use among PhD students as well as to the formation of an army of students attracted by DSGE modeling.

Dynare is a complete and reliable tool that includes a collection of solution methods for the steady state of the model (non-linear models) and uses a local-approximation method around the steady state to solve DSGE models (the perturbation method) based on Klein's solution method. The latest version of Dynare can perform high-order approximations. This software platform can not only solve a calibrated model but also use data to estimate the parameters of a DSGE model, using both maximum likelihood and Bayesian techniques using MCMC methods. Indeed, Dynare has influenced the way in which DSGE models are taken to the data and has expanded the use of estimated DSGE models as opposed to calibration. Dynare includes a high number of options and is able to generate a vast set of results regarding solution, estimation and forecasting with DSGE models. All these elements combined have made Dynare the standard software for those interested in DSGE modeling.

4.3. gEcon

A second specific package is gEcon, developed by Grzegorz Klima, Karol Podemski and Kaja Retkiewicz-Wijtiwiak and presented in 2013. This package was developed in R to take advantage of the high availability of R code for economic and econometric modeling. gEcon is based on a comprehensive symbolic computation library with an object-based R interface. The model can be calibrated (including the use of input–output matrices or social accounting matrices given that CGE models can also be solved).

One particular characteristic of gEcon is that models are written as they are defined. That is, for gEcon, a model is the set of equations comprising the optimization problems of economic agents. Indeed, the main differential characteristic of gEcon is that the model

can be solved directly by writing the optimization problems for different economic agents. The first-order condition from problem maximization, steady state and linearization matrices can be derived through this software. This is different from the definition of the model used by Dynare, in which the model is the set of first-order conditions plus technological, budget and feasibility constraints and state accumulation equations. This means that first-order conditions are not needed when using gEcon as it calculates them itself. Additionally, the process of solving the model is interactive, which means that the values of parameters can be changed without the need to recompute the model. The definition of the model is organized in blocks for each economic agent, for which information on the optimization problem (control variables, objective function, constraints, etc.) or equilibrium relationships is supplied. All this information is written in a *gcn* file, which is read by R using a function that calls on a shared library for symbolic computation, converting the *gcn* file into an R script that solves and simulates the model. Additionally, gEcon can estimate a model using the maximum likelihood approach or Bayesian estimation using the random walk Metropolis–Hastings algorithm.

Another important characteristic of gEcon is that it is a unified computing package for both DSGE and computable general equilibrium (CGE) models. Traditionally, CGE software (GAMS, MPSGE and GEMPACK) has been different from applications for DSGE models, the main reason being that CGE models are designed for static analysis whereas DSGE models are dynamic. However, gEcon is an integrated package that can be used to solve both types of models.

4.4. IRIS

Another package for DSGE modeling is IRIS. IRIS consists of a toolbox for MATLAB that has been developed since 2001 by the IRIS Solutions Team, headed by Jaromir Benes, for macroeconomic modeling and forecasting. IRIS can solve, simulate and estimate (using maximum likelihood methods) DSGE models. Forecasting using the structural model is also possible. Currently, it is supported by the Global Projection Model Network (GPMN). IRIS is a free, open-source MATLAB toolbox, which uses a very flexible language with the possibility of carrying out simulation, estimation, forecasting and model diagnosis. The working of IRIS is similar to that of Dynare, in which the structure of the economic model is described in a model file. Nevertheless, as in the case

of gEcon, the competition of Dynare has limited the use of IRIS as a standard tool by DSGE macroeconomists.

4.5. YADA

YADA (**Y**et **A**nother **D**SGE **A**pplication) is another program for DSGE modeling and structural macroeconometric model estimation, incorporating Bayesian estimation techniques. This code was developed by the New Area-Wide Model (NAWM) team at the Forecasting and Policy Modelling Division of the European Central Bank (ECB) in 2006, using code from other researchers. YADA uses a MATLAB version of the Anderson–Moore (AiM) algorithm for solving linear rational expectations models, but the program also includes other solution algorithms, such as Klein's solution method, as well as *csminwel* (for optimization) and *gensys* (for solving a DSGE model), developed by Christopher Sims. It can also be used to solve models with alternative expectation mechanisms. A distinctive feature compared with other packages, such as Dynare, is that YADA is a GUI-based program, in which the user can control all actions and settings. The latest version of YADA, version 4.90, was released in January 2022 (Warne, 2022). YADA incorporates the code for a number of the most famous DSGE models, produces various results, including the Smets and Wouters (2003) model, a model with a zero lower bound constraint, the DSGE-VAR model and so on and provides a set of Bayesian estimation tools.

4.6. Dolo/Jolo

A more recent source taking advantage of new programming languages is Dolo/Jolo. Dolo/Jolo is a tool or, more specifically, a pre-processor to assist researchers in solving several types of DSGE models, developed by Pablo Winant for Python and Julia. In solving DSGE models, Dolo/Jolo can use either local or global approximation methods. Dolo is a pre-processor for Python, whereas Jolo is a pre-processor for Julia, although later code developed in Julia is also named Dolo.

The logic and main characteristics of this pre-processor are similar to those of Dynare. Users can write the model in a text file YAML (YAML stands for Yet Another Markup Language), in which the components of the models (variables, parameters and equations)

are defined. Sections of a basic file are composed of symbols, equations, calibration, domain and options, with a structure similar to that of the "mod" files of Dynare. Dolo then compiles that text file into Python of Julia. One important characteristic is that Dolo understands several types of non-linear models with occasionally binding constraints, which can be a very useful characteristic for studying some particular economic environments. The software also checks the consistency of the model and computes an efficient numerical representation. The user can choose a particular solution method using one of the methods provided by the package or use one of the already implemented procedures. Currently, only time iteration is supported, but it is expected that, in the near future, value function iteration and parametrized expectations procedures will be available.

4.7. Model comparison: The Macroeconomic Model Data Base

Finally, it is worth mentioning the Macroeconomic Model Data Base (MMB), a project headed by Volker Wieland and supported by a number of contributors (Wieland et al., 2012, 2016). This is a collection of dynamic macroeconomic models based on a common computational platform for systematic model comparison with an application that works with MATLAB/Octave and Dynare. The MMB aims to make a large number of structural macroeconomic models more reproducible, collaborative and comparative. Accordingly, the MMB initiative proposes a unified and straightforward software application for model comparison and result replication. The MMB application collects more than 150 models with the code written in Dynare, including the most popular DSGE models, as a replication source for a number of papers, and a tool for model comparison. The database includes both calibrated and estimated DSGE models for the US, the euro area, multi-country models and other specific-country models as well as a number of models with adaptive learning expectations. The replication package, in which the Dynare code for all models remains as close as possible to the authors' original code or article, can be downloaded from the website (https://www.macromodelbase.com). The MMB interface is divided into five menus—Models (151 models in MMB 3.1.0), Policy Rules (nine monetary policy rules plus one user-specified rule), Shocks (monetary policy shocks, fiscal policy shocks and model specific shocks), Variables (inflation, interest rate, output, output gap and model specific variables) and Options—and it allows the comparison of a model with alternative policy rules or different models for a particular policy rule.

5. Concluding remarks

During the last decades, the progress in macroeconomic theory and that in computing technology have been interconnected, and the current way in which macroeconomic analysis is carried out is strongly related to the advances in computing and programming languages. Computer technology has experienced significant advances during the last century, and it is expected that these will continue or even accelerate in the near future, especially with the transition to quantum computing. Computing advances have allowed the generalization of economic experiments that are difficult to perform otherwise by applying numerical methods and techniques to solve complex theoretical macroeconomic models. This is the main approach followed by macroeconomics, in which the simulation of DSGE models is the core of the macroeconomic laboratory used by researchers and practitioners for policy evaluation.

Kocherlakota (2009) argued that macroeconomic modeling is a by-product of computing technology and that computing technology has been a barrier to the development of more complex DSGE models. The spread of DSGE modeling as the ground of current macroeconomics is related to advances in computers and programming languages and a significant amount of code developed by some leading contributors. The democratization of DSGE access with packages and pre-processors has both positive and negative effects, with the balance clearly in favor of the former. It is clear that this software has eliminated the main barriers to introducing economists to the solution and simulation of DSGE models as little knowledge of programming and numerical methods is required. However, it is true that this is a pure black box and can turn part of macroeconomic modeling into a purely mechanical procedure, similar to that of empirical econometrics, as the cost of solving and simulating DSGE models is currently very small, and it can make a large amount of free code available for a large number of alternative models.

In the near future, it is expected that the importance of computers and software in the development of macroeconomics will increase, allowing the foundation of more advanced, complex and realistic theoretical models. The development of faster and more powerful computers, new code in more advanced programming languages, more advanced and efficient computational numerical methods and more advanced pre-processors will allow the development of DSGE models without the need to use the

representative agent assumption, allowing interactions among a high number of heterogeneous economic agents, which will result in richer and more realistic models, improve the macroeconomic laboratory, contribute to a better understanding of economic phenomena and help policymakers to choose the most appropriate policy instruments to face shocks of different natures. This is a promising and very likely avenue given the observed and expected progress in computing technology and the increasing dependence of macroeconomic analysis on computing.

References

Adelman, I. and Adelman, F. L. (1959). The Dynamic Properties of the Klein–Goldberger Model. Econometrica, 27(4): 596–625.

Aldrich, E. M., J. Fernández-Villaverde, R. A. Gallant and Rubio-Ramírez, J. F. (2011). Tapping the Supercomputer under Your Desk: Solving Dynamic Equilibrium Models with Graphics Processors. Journal of Economic Dynamics and Control, 35(3): 386–393.

Altug, S. (1989). Time-to-Build and Aggregate Fluctuations: Some New Evidence. International Economic Review, 30: 889–920.

Auroba, S. B. and Fernández-Villaverde, J. (2015). A Comparison of Programming Languages in Economics. Journal of Economic Dynamics and Control, 58: 265–273.

Bachmann, S. and Strulik, H. (1992). SoWhat 1.6. A Tool for Easy Simulation of Dynamical System. Manuscript.

Bencivenga, V. R. (1992). An Econometric Study of Hours and Output Variation with Preference Shocks. International Economic Review, 33: 449–471.

Blake, A. P. (2012). DSGE Modeling on an iPhone/iPad Using SpaceTime. Computational Economics, 40: 313–332.

Blanchard, O. J. and Kahn, C. H. (1980). The Solution of Linear Difference Models under Rational Expectations. Econometrica, 48: 1305–1311.

Blanchard, O. J. and Kiyotaki, N. (1987). Monopolistic Competition and the Effects of Aggregate Demand. American Economic Review, 77(4): 647–666.

Bollard, A. E. (2011). Man, Money and Machines: The Contributions of A. W. Phillips. Economica, 78(309): 1–9.

Bongers, A., Gómez, T. and Torres, J. L. (2020). Teaching Dynamic General Equilibrium Models to Undergraduates Using a Spreadsheet. International Review of Economic Education, 35(4): 1–11.

Bongers, A., Gómez, T. and Torres, J. L. (2021). Dynamic Macroeconomic Models with Excel. Journal of Economic Education, 52(4): 372-372.

Brainard, W. C. and Scarf, H. E. (2005). How To Compute Equilibrium Prices in 1891. American Journal of Economics and Sociology, 61(1): 57–83.

Brock, W. and Mirman, L. (1972). Optimal Economic Growth and Uncertainty: The Discounted Case. Journal of Economic Theory, 4(3): 479–513.

Caraiani, P. (2019). Introduction to Quantitative Macroeconomics Using Julia. Academic Press, Elsevier.

Cass, D. (1965). Optimum Growth in an Aggregative Model of Capital Accumulation. Review of Economic Studies, 32: 233–240.

Christiano, L. J. (1988). Why Does Inventory Investment Fluctuate So Much? Journal of Monetary Economics, 21: 247–280.

Chu, A. C. (2018). From Solow to Romer: Teaching Endogenous Technological Change in Undergraduate Economics. International Review of Economics Education, 27(c): 10–15.

DeJong, D. N., Ingram, B. F., and Whiteman, C. H. (2000). A Bayesian Approach to Dynamic Macroeconomics. Journal of Econometrics, 98(2), 203-223.

Dasgupta, P. and Heal, G. (1974). The Optimal Depletion of Exhaustible Resources. Symposium on the Economics of Exhaustible Resources. Review of Economic Studies, 41: 3–28.

Enke, S. (1951). Equilibrium among Spatially Separated Markets: Solution by Electric Analogue. Econometrica, 19(1): 40–47.

Fernández-Villaverde, J. (2010). The Econometrics of DSGE Models. SERIEs, 1(1–2): 3–49.

Forrester, J. W. (1958). Industrial Dynamics—A Major Breakthrough for Decision Makers. Harvard Business Review, 36(4): 37–66.

Goodwin, R. M. (1951). The Nonlinear Accelerator and the Persistence of Business Cycles. Econometrica, 19: 1–17.

Hansen, G. D. (1985). Indivisible Labor and the Business Cycle. Journal of Monetary Economics, 16: 309–327.

Hobbs, P. and Judge, G. (1992). Computers as a Tool for Teaching Economics. Computers and Education, 19 (1-2), 67-72.

Ireland, P. N. (1997). A Small, Structural, Quarterly Model for Monetary Policy Evaluation. Carnegie-Rochester Conference Series on Public Policy, 47, 83-108.

Ireland, P. N. (2001a). Technology Shocks and the Business Cycle. Journal of Economic Dynamics and Control, 25, 703-719.

Ireland, P. N. (2001b). Sticky-price Models of the Business Cycle. Specification and Stability. Journal of Monetary Economics, 47, 3-18.

Ireland, P. N. (2004). A Method for Taking Models to the Data. Journal of Economic Dynamics and Control, 28: 1205–1226.

Jenkins, B. C. (2022). A Python-Based Undergraduate Course in Computational Macroeconomics. Journal of Economic Education, forthcoming.

Judd, K. L. (1997). Computational Economics and Economic Theory: Substitutes or Complements? Journal of Economic Dynamic and Control, 21: 907–922.

Kendrick, D. A. and Amman, H. M. (1999). Programming Languages in Economics. Computational Economics, 14: 151–181.

Kim, J. (2000). Constructing and Estimating a Realistic Optimizing Model of Monetary Policy. Journal of Monetary Economics, 45(2): 329–359.

King, R. G., Plosser, C. I. and Rebelo, S. T. (1987). Production, Growth, and Business Cycles: Technical Appendix. Manuscript. University of Rochester.

Klein, P. (2000). Using the Generalized Schur Form to Solve a Multivariate Linear Rational Expectations Model. Journal of Economic Dynamic and Control, 24: 405–423.

Kocherlakota, N. R. (2009). Modern Macroeconomic Models as Tools for Economy Policy. The Region: 5–21.

Koopmans, T. C. (1965). On the Concept of Optimal Growth, The Econometric Approach to Development Planning. Rand McNally.

Kowal, P. (2005). GEMLLIB-Matlab Code for Specifying and Solving DSGE Models. Computer Programs 0504007. University Library of Munich.

Kydland, F. and Prescott, E. (1982a). Time To Build and Aggregate Fluctuations. Econometrica, 50: 1350–1372.

Kydland, F. and Prescott, E. (1982b). Executable Program for Time To Build and Aggregate Fluctuations. QM&RBC Codes 4, Quantitative Macroeconomics & Real Business Cycles.

Long, J. B. and Plosser, C. I. (1983). Real Business Cycles. Journal of Political Economy, 91(1): 39–69.

Lucas, R. E. (1976). Econometric Policy Evaluation: A Critique. In K. Brunner and A. Meltzer (eds), The Phillips Curve and Labor Markets. Carnegie-Rochester Conference Series on Public Policy 1, New York.

Lucas, R. E. (1980). Methods and Problem in Business Cycle Theory. Journal of Money, Credit, and Banking, 12(4): 696–715.

Mankiw, N. G. (1985). Small Menu Costs and Large Business Cycles: A Macroeconomic Model of Monopoly. Quarterly Journal of Economics, 100: 529–539.

McGrattan, E. (1994). The Macroeconomic Effects of Distortionary Taxation. Journal of Monetary Economics, 33(3): 573–601.

McGrattan, E., Rogerson, R. and Wright, R. (1997). An Equilibrium Model of the Business Cycle with Household Production and Fiscal Policy. International Economic Review, 38(2): 267–290.

Morehouse, N. F., Strotz, R. H. and Horwitz, S. J. (1950). An Electro-analog Method for Investigating Problems in Economic Dynamics: Inventory Oscillations. Econometrica, 18(4): 313–328.

Nerlove, M. (2004). Programming Languages: A Short History for Economists. Journal of Economic and Social Measurement, 29: 189–203.

Neumuller, S., Rothschild, C. and Weerapana, A. (2018). The Macro Pedagogy Debate: Teaching DSGE to Undergraduates Symposium. Journal of Economic Education, 49(3): 242–251.

Perla, J., Sargent, T. J. and Stachurski, J. (2022). Quantitative Economics with Julia. Manuscript.

Phillips, A. W. H. (1950). Mechanical Models in Economic Dynamics. Economica, 17(67): 283–305.

Pierse, R. G. (2000). WinSolve Version 3: An Introductory Guide. Department of Economics, University of Surrey.

Rabanal, P. and Rubio-Ramirez, J. F. (2003). Inflation Persistence: How Much Can We Explain? Economic Review, Federal Reserve Bank of Atlanta, Q2: 43–55.

Ramsey, F. (1928). A Mathematical Theory of Saving. Economic Journal, 37: 543–559.

Raybaut, A. (2020). Analog Computing Simulations and the Production of Theoretical Evidence in Economic Dynamics. Œconomia, 10(2): 309–329.

Rotemberg, J. (1982). Monopolistic Price Adjustment and Aggregate Output. Review of Economic Studies, 49(4): 517–531.

Sargent, T. J. and Stachurski, J. (2015). Quantitative Economics with Julia. Manuscript.

Sergi, F. (2018). DSGE Models and the Lucas Critique. A Historical Appraisal. Working Paper 20181806. Department of Accounting, Economics and Finance, Bristol Business School, University of the West of England, Bristol.

Sims, C. (2001). Solving Linear Rational Expectations Models. Computational Economics, 10: 1–20.

Smets, F. and Wouters, R. (2003). An Estimated Dynamic Stochastic General Equilibrium Model for the Euro Area. Journal of the European Economic Association, 1: 1123–1175.

Solis-García, M. (2018). The Macro Pedagogy Debate: Teaching DSGE to Undergraduates Symposium: Yes We Can! Teaching DSGE Models to Undergraduate Students. Journal of Economic Education, 49(3): 226–236.

Strotz, R. H., Calvert, J. F. and Morehouse, N. F. (1951). Analogue Computing Techniques Applied to Economics. IEEE Xplore, 70: 557–563.

Strotz, R. H., McAnulty, J. C. and Naines, J. B. (1953). Goodwin's Nonlinear Theory of the Business Cycle: An Electro-analog Solution. Econometrica, 21(3): 390–411.

Strulik, H. (1992). SoWhat for Windows 1.6. QM&RBC Codes 96, Quantitative Macroeconomics & Real Business Cycles.

Svensson, L. (1986). Sticky Goods Prices, Flexible Asset Prices, Monopolistic Competition and Monetary Policy. Review of Economic Studies, 53(3): 385–405.

Taylor, J. B. and Uhlig, H. (1990). Solving Nonlinear Stochastic Growth Models: A Comparison of Alternative Solution Methods. Journal of Business and Economic Statistics, 8: 1–17.

Uhlig, H. (1999). A Toolkit for Analyzing Nonlinear Dynamic Model Easily. In R. Marimon and A. Scott (eds), Computational Methods for the Study of Dynamic Economies. New York: Oxford University Press.

Uzawa, H. (1965). Optimum Technical Change in an Aggregative Model of Economic Growth. International Economic Review, 6: 18–31.

Warne, A. (2022). YADA Manual—Computational Details. http://www.texlips.net/yada/

Wieland, V., Afanasyeva, E., Kuete M. and Yoo, J. (2016). New Methods for Macro-Financial Model Comparison and Policy Analysis. Handbook of Macroeconomics, 2: 1241–1319.

Wieland, V., Cwik, T., Müller, G. J., Schmidt, S. and Wolters, M. (2012). A New Comparative Approach to Macroeconomic Modeling and Policy Analysis. Journal of Economic Behavior and Organization, 83: 523–541.