UNIVERSITÀ DEGLI STUDI DI ANCONA
DIPARTIMENTO DI ECONOMIA

MATLAB IMPLEMENTATION OF THE AIM ALGORITHM: A BEGINNER'S GUIDE

Paolo Zagaglia

QUADERNI DI RICERCA n. 169

QUADERNI DI RICERCA

UNIVERSITÀ DEGLI STUDI DI ANCONA
DIPARTIMENTO DI ECONOMIA

MATLAB IMPLEMENTATION OF THE AIM ALGORITHM: A BEGINNER'S GUIDE

Paolo Zagaglia

QUADERNI DI RICERCA n. 169

July 10, 2002

Abstract

The Anderson-Moore algorithm provides a well-established solution method for forward-looking linear rational expectations models. It is widely used at the Federal Reserve Board for a variety of purposes, ranging from simulations of macroeconometric models to computations based on models of monetary policy.

The aim of this paper is to support a wider use of the Anderson-Moore method by discussing the practical sides of its application. I describe the features of one of its Matlab implementations that is freely downloadable from the web. Experience shows that one is usually required to spend quite some time in order to fully understand how the available Matlab functions work. The emphasis is on the structures that should be modified to tailor the programs to one's needs. I also present the application of the algorithm to Coenen and Wieland (2000)'s macromodel of the Euro area.

JEL Class.: C88, E17, C63.
Keywords: macroeconomic modelling, algebraic methods, software.

Indirizzo: Paolo Zagaglia
Centro Studi Luca d'Agliano, Bocconi University
Via Sarfatti, 25 - 20136 Milano
Department of Economics, University of Ancona
P.le Martelli, 8 - 60121 Ancona
E-mail: pzagagli@ra.abo.fi

Matlab Implementation of the AIM Algorithm: A Beginner's Guide*

Paolo Zagaglia

I thank Riccardo Lucchetti for very useful comments. Luca Papi generously allowed me to work full time on my own projects instead of engaging in research assistance. Financial support from Banca Popolare di Ancona through a research grant at the Department of Economics of the University of Ancona is gratefully acknowledged.

Contents

List of Tables

1 Preface

Last March I faced the task of building up a discrete-time stochastic model for another paper. I immediately realized I was forced to include many leads and lags within a rational-expectations structure. The problem of getting a stable solution for my model arose after some time, and the Anderson-Moore - AIM - algorithm emerged as the only procedure that could satisfy my needs. Lacking an adequate network of contacts, I opted for a 'learning-by-doing' approach, and I downloaded the Matlab version of the AIM from its official webpages at the Federal Reserve Board's site. In the remainder of the paper, I will refer to this version as the *alternative* one:

www.bog.frb.fed.us/pubs/oss/oss4/aimindex.html.

It was deeply frustrating to see that a key executable file did not work on my PC, thus endangering the good will of my research project. Luckily, Gary Anderson replied to my desperate call for help, and I am most grateful to him. Gary suggested me to try with a different Matlab implementation, namely one that involves only pure Matlab code. That is downloadable from the webpages hosted by the Federal Reserve Bank of Boston. It is also the version used both on this, and other more renowned papers:

www.bos.frb.org/economic/special/matlab.htm.

At this point, the usual disclaimers are needed. The content of the present paper is the outcome of my scientific interests. Passion has guided me through the mysteries of the AIM algorithm. The use of this guide is intended for educational or research purposes only. Neither the Federal Reserve, nor Gary Anderson himself, nor the authors of the code described herein bear any responsibility for what follows. I accept no liability for either any use of the instructions provided in this document, or the sample code developed to clarify the exposition. As a matter of fairness, I abstain from reporting any line of programs written by other authors. Should any notice of copyright be acknowledged, or proper credit be given to further works in the field, please do not hesitate to contact me at

`paolo.zagaglia@abo.fi`.

2 Introduction

In modern economic research, situations often arise when one needs to compute solutions of alternative models. To this end, the speed, computational efficiency, and flexibility of solution methods represent key features in establishing their usefulness. The algorithm developed in Anderson and Moore (1985) has emerged as a powerful tool for the analysis of rational expectations - RE - models. Most of the work with this method is carried out at the Federal Reserve, and limited knowledge of its functioning is shared by outside economists (Anderson, 2000).

The aim of this paper is to support a widespread use of the AIM algorithm by discussing the practical sides of its implementation. I describe the features of the Matlab package downloadable from the Boston Fed website listed in the preface. The structure of the code is explained 'as it is', although modifications of the original programs are proposed at some point. The paper assumes a somewhat 'intermediate' knowledge by the user concerning Matlab programming, in particular dealing with matrices and functions.

Experience shows that one is usually required to spend quite some time in order to fully understand how the code works. Hence, I believe that what follows can contribute to an appreciation of the virtues of the AIM. In a way, this is the companion paper to Anderson (1999), which is based on the *alternative* Matlab package. Nevertheless, there are several differences with respect to that paper. I avoid dealing with the technicalities involved in the steps of the procedure, relying on intuition rather than formalism. The interested reader can refer to Anderson (2000) for a more thorough exposition. Although the focus is on the solution of RE models, I provide a general overview on a number of connected applications, including estimation and calculations based on the output of the algorithm.

The plan for the paper is as follows. Section 3.1 states the mathematical problem with which this note deals, and section 3.2 outlines the solution strategy proposed by Anderson and Moore (1985). In section 4, I review some computations and estimation techniques that exploit the AIM. Then, I discuss the structure of the Matlab package, and the tasks performed by each function - section 5. Related issues concern the input for the implementation of the AIM algorithm - section 6 -, and the diagnostic output on the stability of the solutions - section 7. Output in terms of key matrices is synthesized in various tables throughout the text. The following section - 8 - presents the application of the algorithm for the solution of a medium-sized macroeconometric model. The conclusions are drawn up in section 9.

3 How the AIM algorithm works

3.1 The Problem

The AIM algorithm is suited for solving structural models with rational expectations expressed as

$$\sum_{i=-\tau}^{0} H_i x_{t+i} + \sum_{i=1}^{\theta} H_i E_t(x_{t+i}) = \varepsilon_t, \quad \tau > 0, \quad \theta > 0. \tag{1}$$

The vector x_t contains all the variables, irrespective of whether they have an endogenous or an exogenous nature. The H_i denote square matrices. I make the assumption that the shock term ε_t follows the normal $N(0,\Omega)$ distribution. Equation 1 is the so-called **structural** representation of the model. Initial **explicit constraints** from data on past observations $\widetilde{x}_t$ can also be imposed:

$$x_\iota = \widetilde{x}_\iota, \quad \iota = -\tau, \ldots -1.$$

The specification of equation 1 can be modified to include constant terms. Two strategies can be followed to handle this extension. The variables can be re-expressed as deviations from steady-state or equilibrium values which are, in turn, given by the constants. The other option consists in adding one equation $\overline{x}$ in the form of a dynamic equality for each constant c:

$$\overline{x}_t = \overline{x}_{t-1},$$

and

$$\overline{x}_{t-1} = c.$$

Without loss of generality, assume that expectations are formed rationally conditional on time t information. After leading the structural representation, and taking expectations, one obtains a homogenous system of forward-looking equations:

$$\sum_{i=-\tau}^{\theta} H_i E_t(x_{t+k+i}) = 0, \quad k \geq 0. \tag{2}$$

I will refer to the H_i's as **structural coefficient** matrices. The method outlined in Anderson and Moore (1985) can then be applied to obtain the solution of equation 2 as a function of the expectations of the past and the present.

A few considerations are needed here. The structural form is general enough to deal with a variety of purely linear models. The use of the AIM does not depend on the degree of backward/forward-lookingness of the equations. An arbitrarily large number of leads and lags can be included. Other solution algorithms impose restrictions already at this stage (see Dennis, 2001). Klein (2000) discusses the use of the generalized Schur decomposition for solving RE models in Vector-Autoregressive - **VAR** - form of order 1 under the assumption that standard regularity conditions apply. In the bulk of papers on inflation targeting produced by Lars Svensson (see Svensson, 1998), forward-looking models are expressed in the state-space form

$$A_0 \begin{bmatrix} x_{1t+1} \\ E_t x_{2t+1} \end{bmatrix} = A_1 \begin{bmatrix} x_{1t} \\ x_{2t} \end{bmatrix} + Bi_t + \begin{bmatrix} \varepsilon_{t+1} \\ 0 \end{bmatrix}.$$

The procedures reviewed in Söderlind (1999) are then employed to study optimal monetary policy.

Finally, one can apply the AIM method to models of whatever size. Examples with forward-looking equations range from small rational-expectations models (see Coenen and Wieland 2000, Fuhrer 1996, Fuhrer 1997a, Fuhrer 1997b, Fuhrer and Madigan 1997, Fuhrer and Moore 1995a, Orphanides 1998, Orphanides and Wieland 1998, Orphanides et al. 1997, Rudebusch 2002), to medium-sized and large models as in Levin et al. (1999, 2001).

3.2 A Sketch of the Solution Procedure

The main task brought carried by the AIM is to find out about the existence of unique or multiple solutions to equation 2. The steps undertaken in the method involve:

1. A full-rank linear transformation of the structural coefficient matrix into a state-space transition matrix A. This generates an autoregressive representation for the law of motion of the state variables, the so-called **unconstraiued autoregression**

$$\begin{bmatrix} x_{-\tau+1} \\ .. \\ x_\theta \end{bmatrix} = \mathrm{A} \begin{bmatrix} x_{-\tau} \\ .. \\ x_{\theta-1} \end{bmatrix}.$$

   Additional sets of **implicit conditions** are also computed.

2. Calculation of the left invariant subspace and the eigenvalues associated with the transition matrix. Blanchard and Kahn (1980) show that, for a system of rational expectations equations to have a unique solution, there must be an adequate number of explosive and stable eigenvalues. Thus, this step focuses on the invariant subspace of large eigenvalues - i.e. those bigger than one. The aim is to recover **convergence constraints** for the trajectories of the system. Anderson and Moore (1985) also demonstrate that these conditions should be linearly independent of both explicit, and implicit constraints in order to generate unique solutions.

3. Computation of **asymptotic constraints** made up of explicit and implicit conditions, and the vectors spanning the invariant subspace.

In order to speed up the computations, sparse matrix methods are used. Depending on the relation between auxiliary conditions and the invariant subspace, the algorithm may rule out in favor of:

- no convergent solutions,
- a unique convergent solution,
- an infinity of convergent solutions.

The AIM algorithm is useful especially when the H_i's of the lead terms turn out to be singular - i.e. they do not have full rank. In this case, the determination of the state-space representation becomes rather problematic. It is possible to find linear combinations of the rows of the structural coefficient matrices that are rank-preserving, and that annihilate the unnecessary rows (see Anderson, 2000). In particular, the AIM exploits a QR decomposition of the singular H_i's to compute the representation

$$H_i = Q_i \cdot R_i,$$

with an orthogonal matrix Q_i and an upper-triangular R_i.[1] After premultiplying H_i by the transpose of Q_i, the algorithm shifts all the non-zero rows to the right. The rows with null elements are instead grouped in the left-upper part of the matrix. Multiplication and 'shift-to-the-right' are iterated until the resulting matrix becomes non-singular. The remaining steps are the same as those outlined earlier.

Should a unique convergent solution arise, the user gets a vector-autoregressive representation of the solution path:

$$E_t(x_{t+k}) = \sum_{i=-\tau}^{-1} B_i E_t(x_{t+k+i}). \quad (3)$$

[1]Other approaches are based on martingale-difference methods (Binder and Pesaran, 1995), system-reduction techniques (King and Watson, 1998), or the generalized Schur decomposition (Klein, 2000).

The B_i's are called **reduced-form coefficients**. The explicit solution for expectations of the future can be plugged into equation 1 so as to compute the **observable structure**:

$$\sum_{i=-\tau}^{0} S_i x_{t+i} = \varepsilon_t. \tag{4}$$

Its denomination is due to the fact that, unlike in the original form 1, no unobserved terms enter equation 4. Furthermore, it is a structural representation of the model, as it is a function of the structural shocks. The observable representation lays the ground for computations as well as estimation of the structural parameters (see section 8).

4 Computations and Estimation based on the AIM: Examples from the Literature

Equation 4 can be pre-multiplied by $-S_0^{-1}$ to obtain the **reduced form** of the structural model:

$$x_t = \sum_{i=-\tau}^{-1} B_i x_{t+i} + B_0 \varepsilon_t. \tag{5}$$

Fuhrer and Moore (1995a) notice that the coefficient matrix B_i in equation 5 is identical to the one in equation 3. The model can then be re-arranged in its companion form:

$$y_t = A y_{t-1} + \varsigma_t, \tag{6}$$

$$y_t = \begin{bmatrix} x_t \\ .. \\ x_{t-\tau+1} \end{bmatrix}, \quad A = \begin{bmatrix} B_{-1} & B_{-2} & B_{-\tau} \\ I & .. & 0 \\ .. & I & 0 \end{bmatrix}, \quad \varsigma_t = \begin{bmatrix} B_0 \varepsilon_t \\ .. \\ 0 \end{bmatrix}.$$

After leading equation 6 k periods, and substituting backward, one obtains the conditional forecast of y_t:

$$y_{t+k} = A^k y_t + \sum_{i=1}^{k} A^{k-1} \varsigma_{t+i}. \tag{7}$$

The conditional variance based on time t information is easily computed by exploiting the uncorrelation property of the disturbance term:

$$\Sigma_{t+k|t} = \sum_{i=0}^{k-1} A^i \Xi (A^i)', \tag{8}$$

with Ξ as the unconditional variance of ς_t.

As pointed out in Anderson (1999), one can exploit the conditional covariance of equation 8 to compute the unconditional variance matrix Σ_0 of y_t:

$$\Sigma_0 = \sum_{i=0}^{+\infty} A^i \Xi (A^i)'. \tag{9}$$

Assuming that the summation in equation 9 converges - i.e. y_t is stationary -, one gets

$$A \Sigma_0 (A)' = \Sigma_0 - \Xi,$$

and by the vec operator

$$\text{vec}(\Sigma_0) = (I - (A \otimes A))^{-1} \text{vec}(\Xi).$$

The computation of the autocovariance function Σ_j of y_t is then straightforward (see Fuhrer and Moore, 1995a):

$$\Sigma_j = A \Sigma_{j-1}, \quad j > 0.$$

As said earlier, model solution through the AIM method often goes hand in hand with estimation of the structural - unobserved - parameters. Fuhrer and Moore (1995a, 1995b) use full information maximum likelihood to estimate alternative specifications of forward-looking wage models on US data. Their procedure runs as follows. Assume that one estimates an empirical linear data-generating process - DGP - for a subset x_e of endogenous variables. The remaining variables x_s are instead driven by structural equations only, whose parameters are to be identified. By stacking the x_e and x_s into a vector x_t, one can cast the model as in the general form 1. The model is then solvable via AIM for a given range of parameters, and the observable representation can be derived. The computation of the likelihood function relies on the observable structure of the model and the explicit constraints.[2]

Coenen and Wieland (2000) investigate the empirical fit of several wage-contracting models for Germany, France and Italy. They estimate structural parameters by indirect inference. This requires obtaining reduced-form solutions of the models over a parameter space in the same fashion as Fuhrer and Moore (1995a, 1995b). The subsequent step consists in generating artificial series for the endogenous variables. Finally, the empirical DGP is fit

[2]Fuhrer and Bleakley (1996) explain the technical background of the econometric investigation of Fuhrer and Moore (1995a, 1995b).

to the artificial data, and simulation-based estimates of the parameters are calculated. These are matched to reduce the difference with the empirical estimates for a feasible range of values.

An important extension to the use of the AIM algorithm consists in the calculation of optimal control rules. Departing from the observable structure of a model, Finan and Tetlow (1999) exploit the Lagrange method for the minimization of the quadratic loss function of a policymaker. The Lagrange multipliers are interpreted as costate variables. This allows the authors to augment the original structural model with the costates. The resulting representation involves matrices that are likely to be singular. The AIM is found to produce fast solutions to the optimal control problem. Finan and Tetlow (1999) illustrate the application of their method on the small sticky-price model of Clarida et al. (1998), and the FRB/US macroeconometric model (see Brayton et al., 1997).[3]

5 The Structure of the Matlab Package

Before dealing with the organization of the programs, I find it proper to outline the sequence of actions a user should put in practice to run the code:

- Modify properly the file solve.m;
- Write down the model equations in a separate ASCII-text file - the model file;
- Write the coefficients and program settings in a set.m file - the setpar file;
- Execute solve.m from the Matlab command window.

In the next section, I will deal with the role and the properties of each file listed above. What matters now is that the execution of solve.m initiates the solution of the model. The tree of files called for is represented in table 1. Each line should be read from left to right, with the columns on the left-hand side indicating the original file. For example, solve.m calls for parse_lin which, in turn, activates strip_equals and so on.

It is noteworthy that not all the Matlab files downloadable from the webpages of the Boston FED are needed for the purpose of solution. Most of the code contributes to the maximum likelihood estimation of the structural coefficients, along the lines developed in Fuhrer and Bleakley (1996).

[3] Sample code is downloadable from www.federalreserve.gov/pubs/feds/1999/199951/199951code.zip.

parse_lin	strip_equals remblnks remblanks convert_top write_cof_m remtabs		
	expand_terms	delete_bad_neighbors find_paren	
	sort_terms	string_repl canonical_replacements	
	find_params	nullify_string rem2blanks	
	find_max_lgld	vec	tschk get_data_ts
space aimerr			
tabit	nmspace mat2char		
aim_eig	numeric_shift exact_shift	shiftright	
	eigensystem build_a copy_w reduced_form		
param_top vibes checkaim obstruct obstruct_t1			

Table 1: Tree of Matlab files stemming from solve.m

5.1 Tasks Performed by the Functions

Before continuing, a deal of rethorics should be settled. In what follows I will often formulate statements with reference to parameters and coefficients. This distinction is of great theoretical interest, although of limited practical use. The Matlab code described here was originally conceived as an essential tool for the estimation of macroeconometric models (see section 8). Some features of the code still lay on such a ground. To that end, the term **parameter** indicates the object of estimation, whereas **coefficient** indicate constant values. For the sake of completeness, this conceptual distinction will be maintained. In the section where I discuss an application of the AIM, I will nevertheless neglect it, and I will use the two terms interchangeably.

solve.m. It sets the AIM key files to solve the model, and it defines the diagnostic output at each stage of the computations. The user is required to define a prefix directory, i.e. the path where Matlab performs the operations. This file interacts with the user by asking for

- a parent directory to be combined with the prefix directory,
- a model file name,
- a coefficients setting - setpar - file name,
- the information set on which the expectations are based - either time t or $t-1$ information.

Further input on demand concerns:

- re-parsing the model or loading existing data,
- setting or estimating parameters.

The output on screen consists in

- summary of input entered at the prompt,
- numerical tolerances used in the computations,
- parameter settings,
- synthetic properties of the state transition matrix and the stability conditions,
- synthetic properties of the roots.

As in every Matlab code, the results from the solution procedure are stored into matrices. The most relevant ones are listed in table 2 on page 13. The reader should keep in mind that each of those - rectangular - matrices stacks square matrices starting from the longest lag of the summations, up to the last lead. For instance, cof regroups the structural coefficient matrices in the following fashion:

$$[\ H_{-\tau}\ \ H_{-\tau+1}\ \ \ldots\ \ H_{\theta}\ ]\,.$$

There are also matrices that are generated to support the execution of the programs (see table 3 on page 14), cell arrays (see table 4), and character arrays (see table 5 on page 15).

parse_lin. This is the so-called model parser. It reads the equations in setpar, prepares a list of endogenous variables, stochastic elements, constants, and parameters. The structure of leads and lags is also synthesized. The aim is to express the original model in the form of equation 2. This task is accomplished by finding the matrices H_i. Output on screen through solve.m includes:

- summary on the lead-lag structure,
- summary of the properties of equations and variables.

strip_equals. It moves a variable to the left-hand side of an equation, changing its sign.

space. It writes blank lines into a matrix.

remblnks. It removes blank characters from a character matrix.

remblanks. Same as remblnks.

convert_top. It generates an index of the parameters collected into a vector.

write_cof_m. It prepares the coefficient matrix and the current parameter setting for evaluation through the rest of the programs.

remtabs. It Sets the output string of remblnks for elaboration by remblanks.

aimerr. This function determines the type of solution, and returns explanatory messages for the outcome.

expand_terms. It performs basic algebraic operations to compute explicit expressions for each equation in the original system.

delete_bad_neighbors. It cleans up the track record of strings so as to avoid confusion between names of variables/parameters with overlapping characters.

find_paren. It checks whether there are mismatched parenthesis in the model file.

sort_terms. It turns a column vector of variables into a row vector for the structural coefficient matrix.

canonical_replacements. Eliminates unnecessary strings created during parsing operations.

tabit. It tabulates the input arguments in a matrix form.

nmspace. It creates a blank matrix of characters.

mat2char. It converts a matrix into a character array.

find_params. It finds the parameters of a linear model.

nullify_string. It replaces a string with spaces.

find_max_lgld. It finds the maximum number of leads and lags.

vec. This is the function version of the Matlab command *vec*.

tschk. It checks whether an object is in time-series format.

get_data_ts. It gets data in time-series form for _DATA type variables.

aim_eig. This is the core function of the package, since it solves a linear RE model.

eigensystem. It computes the eigenvectors and the roots of the left invariant subspace. It also finds the 'big' roots, i.e. those larger than a pre-specified upper bound.

exact_shift. It computes the exact implicit initial constraints.

numeric_shift. It computes the numerical approximation of the implicit conditions.

shiftright. It shifts the rows of an input matrix to the right of a certain number of columns.

build_a. It builds the state-space transition matrix, reducing the number of lags to the minimum.

copy_w. It copies the eigenvectors corresponding to the big roots into the matrix of asymptotic constraints.

reduced_form. It computes the reduced-form matrices B_i's.

param_top. Function required to ease the creation of the parameter vector.

vibes. It computes the size and period of oscillation of the non-zero roots of a system.

Matrix name	Definition	Generating function
cof	H_i	parse_lin
cofb	B_i	reduced_form
scof	S_i	obstruct

Table 2: Key matrices generated by the code

checkaim. It plugs the B_i's into the H_i's to check that the solution solves the model.

obstruct. It calculates the observable structure matrices on the basis of expectations conditional on time t information.

obstruct_t1. Same function as obstruct with reference to period $t-1$ information.

6 The Input

This section discusses some selected features of the files needed to enter the structure of the model for solution. I find it useful to start with the representation of the model equations, since that provides for a better understanding of the subsequent procedures.

6.1 The Model File

The structnre of the model file includes:

- A header - `MODEL` - indicating the name of the model;
- A list of all the variables[4] - `ENDOG` - in the equations;
- A list of equations containing the equation name - `EQUATION` -, its type - `EQTYPE` -, and its functional form - `EQ`.[5]
- A command - `END` - that closes the model representation.

[4]Remember: all the variables are endogenous for the AIM!
[5]The indication of the equation type is requested only in the Matlab versione I describe here, and not in the alternative version.

Matrix name	Generating function	Matrix name	Generating function
E_	solve.m	dopar	solve.m
amp	vibes.m	doparse	solve.m
ans	parse_lin.m	dotm	parse_lin.m
carloc	parse_lin.m	endloc	parse_lin.m
conv_nonlin	setcw0r5.m	epsi	setcw0r5.m
eqtype_	parse_lin.m	nex	aim_eig.m
equlocs	parse_lin.m	next	parse_lin.m
err	chackaim.m	nlag	find_max_lgld.m
fid	parse_lin.m	nlead	find_max_lgld.m
first5	parse_lin.m	nnum	aim_eig.m
i	parse_lin.m	np	parse_lin.m
ia	aim_eig.m	nres	setcw0r5.m
ii	parse_lin.m	nvar	parse_lin.m
II	parse_lin.m	oldE_	solve.m
last5	parse_lin.m	p	solve.m
lgrts	aim_eig.m	per	vibes.m
loadflg	solve.m	pinit	solve.m
mcode	aim_eig.m	prnt	solve.m
neq	parse_lin.m	q	checkaim.m
neqc	setcw0r5.m	rts	aim_eig.m
typlocs	parse_lin.m	uprbnd	setcw0r5.m

Table 3: Other matrices

Array	Generating function	Array	Generating function
Hmat	parse_lin.m	param_	parse_lin.m
Hvec	sort_terms.m	terms	expand_terms.m
Hvec2	sort_terms.m	eqname_	parse_lin.m
endog_	parse_lin.m		

Table 4: Cell arrays

Array	Generating function	Array	Generating function
dirnam	solve.m	modnam	solve.m
eqns	parse_lin.m	modname	solve.m
errstr	aimerr.m	namstr	parse_lin.m
mod	parse_lin.m	olddirnam	solve.m
oldmodnam	solve.m	tempeqn	strip_equals.m
oldparnam	solve.m	tempeqns	parse_lin.m
prefdir	solve.m	typstr	parse_lin.m
ptab	tabit.m	vtab	parse_lin.m
str	parse_lin.m		

Table 5: Character arrays

The syntax used in the model file follows the so-called MDLEZ language. All the commands but `END` must be followed by the sign >.

In the `ENDOG` listing, the name of each variable is accompanied to its type. Three kinds of endogenous variables are allowed, denoted as `_DATA`, `_NOTD`, and `_DTRM`. This distinction makes a practical sense in the context of model estimation, with the `_DATA` variables being assigned time series values. A special case is represented by the pseudo-variable one, which is the only one of type `_DTRM`. It is the outcome of a trick which allows the user to include stochastic shocks to the equations. For instance, assume I have added a disturbance `yDE_` to the equation `yDE`. If the shock had a stochastic nature, it would be written in MDLEZ as

$$\texttt{yDE\_} = 0 * \texttt{one}.$$

The equation of `one` would instead have the form

$$\texttt{one} = \mathrm{LAG}(\texttt{one}, 1),$$

The variable `one` enters tbe model as an identity, in the sense that its current value equals its first lag.

The `EQTYPE` contemplates the strings `IMPOSED` - for deterministic equations - and `STOCH` - for stochastic equations. Here a constraint ou the form of the model is imposed, as the number of `_DATA` variables must equal the number of `STOCH`. Again, this is needed for estimation only, since it allows the calculation of the Jacobian matrix of the likelihood function (see section 8). Interestingly, an equation that includes a stochastic shock should be classified as `IMPOSED`. Only the representation of the disturbances themselves are `STOCH`. Finally, the equation for `one` is of type `IMPOSED`.

MDLEZ syntax	vtype_
_DATA	0
_NOTD	1
_DTRM	2

Table 6: Correspondence between variables in MDLEZ and syntax in vtype_

Writing equations is rather intuitive is MDLEZ. The key task is to cope with backward- and forward-looking terms as, respectively,

$$\mathrm{LAG}(\pi, \rho), \quad \text{and} \quad \mathrm{LEAD}(\pi, \varrho),$$

with π the endogenous variable, ρ the number of periods backward, and ϱ the number of periods forward for the expectations. A numeric constant c can instead be included in an equation by entering the coefficient times one.

6.2 The File solve.m

The first issue the user faces consists in dealing with the files location. As noticed in the previous section, the prefix directory can be different from the path requested by the program at the prompt once solve.m is executed. The setpar and model files can be located in either of the two, whereas the rest of the Matlab programs should be in the prefix directory. It is noteworthy that a model parser array and a function both with the name of the model file - but a different extension - are generated by the parser, and saved in the directory entered at the prompt.

The names of endogenous variables and parameters are stored, respectively, into the matrices **endog_** and **param_** through tbe `parse_lin.m`. There are also equation names **eqname_** and types **eqtype_**. Types of equations aud variables are instead defined as **eqtype_** and **vtype_**. Table 6 reports the translation of types of variables from MDLEZ into **vtype_**. The dimensions of the model are collected as **neq** - number of equations - and **nlead** and **nlag** - number of leads and lags. The number of variables is **nvar**, whereas the number of parameters is **np**.

6.3 The File set.m

The main task brought about by this function consists in storing the parameters/constants values to be included in the model equations. Other options of interest include **condn** and **uprbnd**, which are both used by **aim_eig**.

The former is a tolerance number used to compute the left invariant subspace, whereas the latter is an upper bound for the modulus of the roots in the reduced form.[6]

7 Diagnostic Output on Screen

As said earlier, solve.m reports a synthetic description on screen of the solution stability. Since that is crucial for a sound understanding of the results, I dedicate some attention to that point.

The number of - both explicit and implicit - auxiliary constraints is captured by **nex** and **nnum**. Their denominations within the functions that generate them are reported in the central column of table 7. The AIM algorithm makes nse of approximation methods to perform the calculations on the invariant subspace. This is the reason for **nex** to report the constraints identified by symbolic algebra computations, and **nnm** to complement them by numeric algebra. Finally, **lgroots** indicates the number of roots larger than the prespecified **uprbnd**.

On screen	Indicator	Generating function
nex	nexact	exact_shift.m
nnum	nnumeric	numeric_shift.m
lgrts	lgroots	eigensystem.m
ia	ia	aim_eig.m
err	err	checkaim.m

Table 7: Diagnostics of system stability

In order to what kind of solution exists, the AIM compares the number of stability conditions discovered 'along the way' with the number of constraints required. The former is the sum between **nex**, **nnum** and **lgroots**, whereas the latter is obtained as the product of **neq** with **nlead**. Although intuitively close, this way of characterizing the solutions is rather different from the one developed in Blanchard and Kahn (1980), who rely on a comparison between the number of eigenvalues outside the unit circle and the number of non-predetermined variables.

Tbe code also shows a message describing the type of solution obtained. If the number of stability conditions identified is larger than the number

[6] I would suggest the user not to modify the default values before due practice with the AIM has been developed.

of stability conditions required by the model, the used obtains unstable solutions. The code still calculates the coefficients in the case that the QR decomposition is able to find a non-singular orthogonal matrix. The same considerations applies when there are multiple solutions, i.e. there are more auxiliary conditions than what the model requires. Additional output consists in the dimension **ia** of the transition matrix A, as well as the maximum absolute error **err** of the solution (see table 7).

8 An Application: CW's Macromodel of the Euro Area

Other papers explaining the operation of the AIM use small models that are analytically tractable. For instance, Anderson and Moore (1985) is based on the two-equation Harrod-Domar model of economic growth, while Anderson (2000) deals with Sims (1996)'s example of wage contracting with three equations. Since my attention is on 'practical' policy analysis, I opt for a complicated application with a relatively large number of state variables. In this sense, the macroeconometric model developed in Coenen and Wieland (2000) - CW - appears as a good deal.

Although the empirical investigation of CW involves much broader results, I include the equations that appear to exhibit the best statistical fit. The model merges a supply with a demand side for Germany, France and Italy, indexed respectively by ι as 1, 2 and 3. The variables are defined on a quarterly frequency.

The inflation rate is

$$\pi_t^\iota = p_t^\iota - p_{t-1}^\iota.$$

The price level p_t evolves according to a one-year weighted average of nominal wages w_t in each country (see equation 10). The weights f_i^ι are assumed to be downward-sloping functions of the contract length i. In Coenen and Wieland (2000), the nominal-wage contracting - i.e. Taylor's - model appears to fit both the German and French data rather well (see equations 11-14), as these are characterized by stickiness in the price level only. The Italian inflation process exhibits a larger degree of sluggishness, which is accounted for by a version of Fuhrer and Moore (1995a)'s model where historically-available information plays a key role (see equations 15-17).

Equations 18-20 show that the deviation of output from trend y_t^ι is a function of the output gap in the two previous quarters, and the *ex-ante* real interest rate $r_t^{\iota l}$. Owing to equation 23, the demand-side impact of the real

rate of interest is country-specific, as inflation expectations over the following two quarters are likely to diverge. Nominal long-term interest rates arise from the term-structure relation 22, for which the expectations hypothesis holds and the term premium is null. Monetary policy takes the form of a Taylor-type rule driven by area-wide conditions. In equation 21,

$$\tilde{y}_t = \sum_{\iota=1}^{3} s_\iota y_t^\iota \quad \text{and} \quad \tilde{\pi}_t = \frac{1}{4}\sum_{j=0}^{3}\sum_{\iota=1}^{3} s_\iota \pi_{t-j} = \sum_{\iota=1}^{3} s_\iota (p_t^\iota - p_{t-4}^\iota),$$

where the latter is the four-quarter moving average of the annualized quarterly inflation rate. The country weights s_ι are computed according to the ECB Area-Wide Database of Fagan et al. (2001).

8.1 Additional Computations on the CW Model

In order to order to illustrate the flexibility of the code, I calculate the conditional covariance of y_t based on $t-1$ information. Thus, I lag equation 6 one more period. After substituting the resulting expression into itself for many times, I obtain

$$y_{t-1+k} = A^{k+1} y_{t-2} + \sum_{i=0}^{k} A^{k-i} \varsigma_{t-1+i}, \tag{24}$$

Since ξ_t is serially uncorrelated, its conditional variance-covariance matrix is

$$\Sigma_{t-1+k|t-1} = \sum_{i=0}^{k} A^i \Xi (A^i)', \quad \text{and} \quad \Xi = \begin{bmatrix} B_0 \Omega B_0' & 0 & .. \\ 0 & .. & .. \\ .. & .. & 0 \end{bmatrix}, \tag{25}$$

with Ξ as the unconditional covariance matrix of ς_t.

8.2 Comments on the Sample Code

The interpretation of the code listing for the model file Cw0r5 is rather straightforward (see Appendix A). For what concerns setcw0r5.m, the reader should be aware of the fact that I have deleted the lines from the original file needed to perform maximum likelihood estimation. I have also included the values of four constants as parameters. This does not affect the results, since for solution purposes one need not make the econometric distinction between coefficients and parameters. Other constants are in the model file. The **condn** and **uprbnd** are set as 'default' values from the file originally downloaded.

Supply side.

$$p_t^\iota = f_0^\iota w_t^\iota + f_1^\iota w_{t-1}^\iota + f_2^\iota w_{t-2}^\iota + f_3^\iota w_{t-3}^\iota, \quad \iota = 1, 2, 3 \tag{10}$$

$$w_t^1 = E_t \left[\sum_{i=0}^{3} f_i^1 p_{t+i}^1 + 0.0195 \sum_{i=0}^{3} f_i^1 y_{t+i}^1 \right] + u_t^{1w} \tag{11}$$

$$f_i^1 = 0.25 + (1.5 - i)0.0501, \quad u_t^{1w} \sim \text{i.i.d.}(0, 0.0074) \tag{12}$$

$$w_t^2 = E_t \left[\sum_{i=0}^{3} f_i^2 p_{t+i}^2 + 0.0041 \sum_{i=0}^{3} f_i^2 y_{t+i}^2 \right] + u_t^{2w} \tag{13}$$

$$f_i^2 = 0.25 + (1.5 - i)0.1189, \quad u_t^{2w} \sim \text{i.i.d.}(0, 0.0048) \tag{14}$$

$$v_t^3 = \sum_{i=0}^{3} f_i^3 \left(w_{t-i}^3 - E_t[\overline{p}_{t-i}^3] \right), \quad \overline{p}_t^3 = \sum_{i=0}^{3} f_i^3 p_{t+i}^3 \tag{15}$$

$$w_t^3 - E_t[\overline{p}_t^3] = E_t \left[\sum_{i=0}^{3} f_i^3 v_{t+i}^3 + 0.0046 \sum_{i=0}^{3} f_i^3 y_{t+i}^3 \right] + u_t^{3w} \tag{16}$$

$$f_i^3 = 0.25 + (1.5 - i)0.1244, \quad u_t^{3w} \sim \text{i.i.d.}(0, 0.0023) \tag{17}$$

Real demand.

$$y_t^1 = 0.0012 + 0.7865 y_{t-1}^1 + 0.1395 y_{t-2}^1 - 0.0365 r_t^{1l} + u_t^{1d}, \quad u_t^{1d} \sim \text{i.i.d.}(0, 0.0012) \tag{18}$$

$$y_t^2 = 0.0024 + 1.2247 y_{t-1}^2 - 0.2708 y_{t-2}^2 - 0.0638 r_t^{2l} + u_t^{2d}, \quad u_t^{2d} \sim \text{i.i.d.}(0, 0.0003) \tag{19}$$

$$y_t^3 = 0.0023 + 1.3524 y_{t-1}^3 - 0.3852 y_{t-2}^3 - 0.0544 r_t^{3l} + u_t^{3d}, \quad u_t^{3d} \sim \text{i.i.d.}(0, 0.0004) \tag{20}$$

Interest rates.

$$i_t^s = \alpha_r i_{t-1}^s + \alpha_\pi (\widetilde{\pi}_t - \pi^*) + \alpha_y \widetilde{y}_t \tag{21}$$

$$i_t^l = E_t \left[\frac{1}{8} \sum_{j=0}^{7} i_{t+j}^s \right] \tag{22}$$

$$r_t^{\iota l} = i_t^l - E_t \left[\frac{1}{2} (p_{t+8}^\iota - p_t^\iota) \right], \quad \iota = 1, 2, 3 \tag{23}$$

In solve.m, the prefix directory is /matlabr11/aim/. I have placed Cw0r5 and setcw0r5.m into /windows/desktop/project/. The last few lines of the code compute the conditional covariance matrix along the lines discussed in the previous subsection. It is assumed that the unconditional covariance of the shocks - Omega - is an identity matrix. The conditional covariance of the state variables is finally coded as the matrix Sum_part.

8.3 Comments on the Execution of the Code

The programs ran on a PC with a Pentium 75 processor and 64 MB of RAM, i.e. a slow machine. I used Matlab v. 5.3. The execution of the entire procedure took about one minute.

The diagnostic output shows no particular problems (see Appendix B). The model fulfills the conditions to obtain a unique solution. The maximum absolute error of the approximate solution is rather small. As an example, the code computes the covariance $\Sigma_{t+1|t-1}$ at time $t+1$ conditional on time $t-1$ information, i.e. $k = 2$ in equation 25 on page 19.

9 Concluding Remarks

This paper deals with the practical application of the AIM algorithm. In particular, I have discussed the organizing principles of a Matlab package designed to solve linear rational expectations models with forward-looking components. In addition to reviewing the structure of the Matlab functions, I discuss their application to a laboratory macromodel of the Euro area.

The Matlab implementation of the AIM algorithm is capable of finding rather accurate solutions through a combination of algebraic and numerical routines. The execution of the procedure is also rather quick.

The virtues of the AIM are somewhat counterbalanced by the problems of more general nature of approximate solution methods. In particular, notwithstanding the recent advances documented in Anderson (1999), the method is still unable to distinguish between unit roots and near-unit roots. A useful extension on which I am currently working consists in using the econometric results from fractionally-integrated models to assess undetected unstable solutions through the AIM.

A Sample Code

File Cw0r5.

```
MODEL > cw0

ENDOG >
        yDE      _DATA
        yFR      _DATA
        yIT      _DATA
        piDE     _NOTD
        piFR     _NOTD
        piIT     _NOTD
        pDE      _NOTD
        pFR      _NOTD
        pIT      _NOTD
        wDE      _DATA
        wFR      _DATA
        wIT      _DATA
        vIT      _NOTD
        is       _NOTD
        rlDE     _NOTD
        rlFR     _NOTD
        rlIT     _NOTD
        yDE_     _NOTD
        yFR_     _NOTD
        yIT_     _NOTD
        wDE_     _NOTD
        wFR_     _NOTD
        wIT_     _NOTD
        one      _DTRM

EQUATION > yDE
EQTYPE >    IMPOSED
EQ >        yDE =
           .7865  * LAG(yDE,1)
         + .1395  * LAG(yDE,2)
         - .0365  * rlDE
         + .0012  * one
         +      yDE_
```



```
EQUATION > yFR
EQTYPE >    IMPOSED
EQ >        yFR =
           1.2247  * LAG(yFR,1)
         -  .2708  * LAG(yFR,2)
         -  .0638  * rlFR
         +  .0024  * one
         +       yFR_

EQUATION > yIT
EQTYPE >    IMPOSED
EQ >        yIT =
           1.3524  * LAG(yIT,1)
         -  .3852  * LAG(yIT,2)
         -  .0544  * rlIT
         +  .0023  * one
         +            yIT_

EQUATION > piDE
EQTYPE >    IMPOSED
EQ >        piDE = pDE - LAG(pDE,1)

EQUATION > piFR
EQTYPE >    IMPOSED
EQ >        piFR = pFR - LAG(pFR,1)

EQUATION > piIT
EQTYPE >    IMPOSED
EQ >        piIT = pIT - LAG(pIT,1)

EQUATION > pDE
EQTYPE >    IMPOSED
EQ >        pDE =
           .32305 * wDE
         + .27435 * LAG(wDE,1)
         + .22565 * LAG(wDE,2)
         + .17965 * LAG(wDE,3)

EQUATION > pFR
EQTYPE >    IMPOSED
EQ >        pFR =
```

```
         .42835 * wFR
       + .30945 * LAG(wFR,1)
       + .19055 * LAG(wFR,2)
       + .07165 * LAG(wFR,3)

EQUATION > pIT
EQTYPE >   IMPOSED
EQ >       pIT =
         .4366 * wIT
       + .3122 * LAG(wIT,1)
       + .1878 * LAG(wIT,2)
       + .0634 * LAG(wIT,3)

EQUATION > wDE
EQTYPE >   IMPOSED
EQ >       wDE =
         .32305 * pDE
       + .27435 * LEAD(pDE,1)
       + .22565 * LEAD(pDE,2)
       + .17695 * LEAD(pDE,3)
       + .00195 * (.32305*yDE + .27435*LEAD(yDE,1)
       + .22565*LEAD(yDE,2) + .17695*LEAD(yDE,3))
       + wDE_

EQUATION > wFR
EQTYPE >   IMPOSED
EQ >       wFR =
         .42835 * pFR
       + .30945 * LEAD(pFR,1)
       + .19055 * LEAD(pFR,2)
       + .07165 * LEAD(pFR,3)
       + .0041 * (.42835*yFR + .30945*LEAD(yFR,1)
       + .19055*LEAD(yFR,2) + .07165*LEAD(yFR,3))
       + wFR_

EQUATION > wIT
EQTYPE >   IMPOSED
EQ >       wIT - .4366*pIT - .3122*LEAD(pIT,1)
       - .1878*LEAD(pIT,2) - .0634*LEAD(pIT,3)
       = .4366*vIT + .3122*LEAD(vIT,1)
       + .1878*LEAD(vIT,2) + .0634*LEAD(vIT,3)
```



```
       + .0046*.4366*yIT + .0046*.3122*LEAD(yIT,1)
       + .0046*.1878*LEAD(yIT,2) + .0046*.0634*LEAD(yIT,3)
       + wIT_

EQUATION > vIT
EQTYPE >   IMPOSED
EQ >       vIT =
         .4366 * (yIT - .4366*pIT - .3122*LEAD(pIT,1)
         - .1878*LEAD(pIT,2) - .0634*LEAD(pIT,3))
       + .3122 * (LAG(yIT,1) - .4366*LAG(pIT,1)
       - .3122*pIT - .1878*LEAD(pIT,1) - .0634*LEAD(pIT,2))
       + .1878 * (LAG(yIT,2) - .4366*LAG(pIT,2)
       - .3122*LAG(pIT,1) - .1878*pIT - .0634*LEAD(pIT,1))
       + .0634 * (LAG(yIT,3) - .4366*LAG(pIT,3) - .3122*LAG(pIT,2)
       - .1878*LAG(pIT,1)  - .0634*pIT)

EQUATION > is
EQTYPE >   IMPOSED
EQ >       is =
         alphar  * LAG(is,1)
       + alphapi * (.4248*(pDE - LAG(pDE,4))
       + .2922*(pFR - LAG(pFR,4)) + .2829*(pIT - LAG(pIT,4)) - pistar)
       + alphay  * (.4248*yDE + .2922*yFR + .2829*yIT)

EQUATION > rlDE
EQTYPE >   IMPOSED
EQ >       rlDE =
         (1/8) * is
       + (1/8) * LEAD(is,1)
       + (1/8) * LEAD(is,2)
       + (1/8) * LEAD(is,3)
       + (1/8) * LEAD(is,4)
       + (1/8) * LEAD(is,5)
       + (1/8) * LEAD(is,6)
       + (1/8) * LEAD(is,7)
       - (1/2) * LEAD(pDE,8)
       + (1/2) * pDE

EQUATION > rlFR
EQTYPE >   IMPOSED
EQ >       rlFR =
```

```
          (1/8) * is
        + (1/8) * LEAD(is,1)
        + (1/8) * LEAD(is,2)
        + (1/8) * LEAD(is,3)
        + (1/8) * LEAD(is,4)
        + (1/8) * LEAD(is,5)
        + (1/8) * LEAD(is,6)
        + (1/8) * LEAD(is,7)
        - (1/2) * LEAD(pFR,8)
        + (1/2) * pFR

EQUATION > rlIT
EQTYPE >   IMPOSED
EQ >       rlIT =
          (1/8) * is
        + (1/8) * LEAD(is,1)
        + (1/8) * LEAD(is,2)
        + (1/8) * LEAD(is,3)
        + (1/8) * LEAD(is,4)
        + (1/8) * LEAD(is,5)
        + (1/8) * LEAD(is,6)
        + (1/8) * LEAD(is,7)
        - (1/2) * LEAD(pIT,8)
        + (1/2) * pIT

EQUATION > yDE_
EQTYPE >   STOCH
EQ >       yDE_ = 0 * one

EQUATION > yFR_
EQTYPE >   STOCH
EQ >       yFR_ = 0 * one

EQUATION > yIT_
EQTYPE >   STOCH
EQ >       yIT_ = 0 * one

EQUATION > wDE_
EQTYPE >   STOCH
EQ >       wDE_ = 0 * one
```



```
EQUATION > wFR_
EQTYPE >   STOCH
EQ >       wFR_ = 0 * one

EQUATION > wIT_
EQTYPE >   STOCH
EQ >       wIT_ = 0 * one

EQUATION > one
EQTYPE >   IMPOSED
EQ >       one = LAG(one,1)

END
```

File setcw0r5.m.

```
%% ----------------------------------------------
%% Parameter setting for the Coenen-Wieland model
%% ----------------------------------------------

alphar = 0.25;
alphapi = 0.25;
pistar = 2;
alphay = 0.5;

%% ----------------------------------------------

np = length(param_);

for i = 1:np
    p(i) = eval(param_{i});
end

% Set initial parameter values in pinit

pinit = p;

% Set numerical tolerances
```

```
condn = 1.e-8;
uprbnd = 1 + 1.0e-6;
```

File solve.m.

```
% #####################
%  Set Prefix Directory
% #####################

%prefdir = '/YOUR PREFIX DIRECTORY HERE/';
prefdir = '/matlabr11/aim/';

clear p pinit hess numhess thess

% ----------------------------------------------------------------------
% Parameter values, names, endog_ list, model dimensions, and eqtype_,
% vtype_ are defined by parser.  Parameters are set in setpar.
% ----------------------------------------------------------------------

% Request model and parameter file information

dirnam = input('Directory name: ','s');
if(isempty(dirnam))
  if(exist('olddirnam'))
      dirnam = olddirnam;
  else
      disp('No directory name currently defined')
      return
  end
end
if (dirnam(1)=='/')
    prefdir = '';
else
%prefdir = '/YOUR PREFIX DIRECTORY HERE/';
    prefdir = '/matlabr11/aim/';
end

modnam = input('Model file name: ','s');
if(isempty(modnam))
```



```
  if(exist('oldmodnam'))
      modnam = oldmodnam;
  else
      disp('No model name currently defined')
      return
  end
end

parnam = input('Setpar file name: ','s');
if(isempty(parnam))
  if(exist('oldparnam'))
      parnam = oldparnam;
  else
      disp('No paramter program name currently defined')
      return
  end
end

E_ = input('t (0) or t-1 (1) period expectations? ');
if(isempty(E_))
  if(exist('oldE_'))
      E_ = oldE_;
  else
      error('No expectation date currently defined')
  end
end

eval(['cd ',prefdir,dirnam])

olddirnam = dirnam;
oldmodnam = modnam;
oldparnam = parnam;
oldE_ = E_;

% (1) Parse model, if required

doparse = input('Re-parse model? (1=yes)  ');
if(isempty(doparse))
    doparse = 0;
end
if(doparse & exist([prefdir,dirnam,'/',modnam,'_parse.mat']) )
```

```
    loadflg = input('Re-parse (1) or load existing model data (0)? ');
    if(isempty(loadflg))
        loadflg = 0;
    end
else
    loadflg=1;
end

dopar = input('Set parameters? (1=yes)  ');
if(isempty(dopar))
   dopar = 0;
end

if(doparse)
   if loadflg
    parse_lin
   else
    eval(['load ',prefdir,dirnam,'/',modnam,'_parse'])
   end
end

% (2) Define parameters

if(dopar)
  eval(parnam);
  % Set p-vector equal to values of parameters found in param_

  p = [];
  param_top
  pinit = p;
end
if(~exist('p')) p = zeros(1);end

prnt = 0;  % Intermediate output switch


% Display stuff

space
disp('----------------------------------------------------------------------')
disp(['Solving model  :  ',modnam])
```



```
disp(['Model directory:  ',prefdir,dirnam])
disp(['Parameter file :  ',parnam,'.m'])
if(E_==0)
 disp('Expectations viewpoint:     period t.')
elseif(E_==1)
 disp('Expectations viewpoint:     period t-1.')
end
disp('----------------------------------------------------------------------')


% Tolerances

space;
disp('---------------------');
disp('Numerical Tolerances');
disp('---------------------');
disp([' condn     : ',num2str(condn)]);
disp([' uprbnd - 1: ',num2str(uprbnd-1)]);
space;

if(np>0)

space(2)
disp('---------------------')
disp('Parameter settings')
disp('---------------------')
space
disp('Name       Value    ')
disp('                    ')
ptab = tabit(param_,p);
disp(ptab)
space(2)

end

%--------------------------------------------------------------------
% This stuff does aim, obstruct to provide structure for simulation
%--------------------------------------------------------------------

% ------------------------------------------------------------------
% Construct cof matrix using p.
```

```
% ------------------------------------------------------------------

cof = feval([modnam,'_cof'],p);

% ----------------------------------------------------------------
% Run AIM
% ----------------------------------------------------------------

[cofb,rts,ia,nex,nnum,lgrts,mcode] = ...
       aim_eig(cof,neq,nlag,nlead,condn,uprbnd);


disp(['Number of exact shiftrights (nex):    ',num2str(nex)]);
disp(['Number of numeric shiftrights (nnum): ',num2str(nnum)]);
disp(['Number of large roots (lgrts):        ',num2str(lgrts)]);
disp(['Number of stability conditions (nex + nnum + lgrts) -'])
disp(['number required (neq*nlead) = ',num2str(nex+nnum+lgrts-neq*nlead)]);
disp(['Dimension of state transition matrix (ia): ',num2str(ia)]);
errstr = aimerr(mcode);
disp(errstr);


% ----------------------------------------------------------------------
% Display roots, magnitude of roots and period
% ----------------------------------------------------------------------

[amp,per] = vibes(rts,0);

% ----------------------------------------------------------------------
% Check accuracy of solution
% ----------------------------------------------------------------------

[q,err] = checkaim(neq,nlag,nlead,cof,cofb);

% ----------------------------------------------------------------------
% Compute observable structure
% ----------------------------------------------------------------------

if(E_==0)
    scof = obstruct(cof,cofb,neq,nlag,nlead);
elseif(E_==1)
```



```
    scof = obstruct_t1(cof,cofb,neq,nlag,nlead);
else
    error('Improper spec. of expectations viewpoint.')
end


% ------------------------------------------------------------------------
% Calculation of the conditional covariance matrix
%
% Step 1: Form the companion matrix
% ------------------------------------------------------------------------

dimens = size(cofb);
r = [ dimens(1,2)/neq ];

% Coefficient matrix of the VAR system in companion form

A = [ fliplr(cofb);
      eye([r-1]*neq), zeros([r-1]*neq, neq) ];
% notice that the reduced form starts with the longest lag!!

% Define the number of periods ahead in time

k = input('The conditional forecast refers to (# of periods ahead): ');
if(isempty(k))
   k = 2;
end

% Define the unconditional covariance matrix

dimens1 = size(A);
Omega = eye(dimens1(1,2));

% Compute the conditional covariance matrix

Sum_part = zeros(dimens1(1,2));
for i = 0:k,
    C = (A^i)*(Omega)*[(A^i)'];
    Sum_part = C + Sum_part;
end
```

B Output from the Sample Code

```
>> solve
Directory name: /windows/desktop/project
Model file name: cw0r5
Setpar file name: setcw0
t (0) or t-1 (1) period expectations? 0
Re-parse model? (1=yes)  1
Re-parse (1) or load existing model data (0)? 1
Set parameters? (1=yes)  1

Parsing model cw0r5 ...

    Parsing Equation 1: yDE
    Parsing Equation 2: yFR
    Parsing Equation 3: yIT
    Parsing Equation 4: piDE
    Parsing Equation 5: piFR
    Parsing Equation 6: piIT
    Parsing Equation 7: pDE
    Parsing Equation 8: pFR
    Parsing Equation 9: pIT
    Parsing Equation 10: wDE
    Parsing Equation 11: wFR
    Parsing Equation 12: wIT
    Parsing Equation 13: vIT
    Parsing Equation 14: is
    Parsing Equation 15: rlDE
    Parsing Equation 16: rlFR
    Parsing Equation 17: rlIT
    Parsing Equation 18: yDE_
    Parsing Equation 19: yFR_
    Parsing Equation 20: yIT_
    Parsing Equation 21: wDE_
    Parsing Equation 22: wFR_
    Parsing Equation 23: wIT_
    Parsing Equation 24: one

Done.

MODEL:   cw0
```



```
Number of equations: 24
Number of lags      : 4
Number of leads     : 8

Endog.         Variable        Equation Equation
Var.            Type            Name      Type

---------------------------------------
yDE     0          yDE     1
yFR     0          yFR     1
yIT     0          yIT     1
piDE    1          piDE    1
piFR    1          piFR    1
piIT    1          piIT    1
pDE     1          pDE     1
pFR     1          pFR     1
pIT     1          pIT     1
wDE     0          wDE     1
wFR     0          wFR     1
wIT     0          wIT     1
vIT     1          vIT     1
is      1          is      1
rlDE    1          rlDE    1
rlFR    1          rlFR    1
rlIT    1          rlIT    1
yDE_    1          yDE_    0
yFR_    1          yFR_    0
yIT_    1          yIT_    0
wDE_    1          wDE_    0
wFR_    1          wFR_    0
wIT_    1          wIT_    0
one     2          one     1
---------------------------------------


Parameters

----------------------
 1          alphar
 2          alphapi
 3          pistar
 4          alphay
```

```
----------------------

Equations:
-----------

(1) yDE=.7865*LAG(yDE,1)+.1395*LAG(yDE,2)-.0365*rlDE+.0012*one+yDE_

(2) yFR=1.2247*LAG(yFR,1)-.2708*LAG(yFR,2)-.0638*rlFR+.0024*one+yFR_

(3) yIT=1.3524*LAG(yIT,1)-.3852*LAG(yIT,2)-.0544*rlIT+.0023*one+yIT_

(4) piDE=pDE-LAG(pDE,1)

(5) piFR=pFR-LAG(pFR,1)

(6) piIT=pIT-LAG(pIT,1)

(7) pDE=.32305*wDE+.27435*LAG(wDE,1)+.22565*LAG(wDE,2)+.17965*LAG(wDE,3)

(8) pFR=.42835*wFR+.30945*LAG(wFR,1)+.19055*LAG(wFR,2)+.07165*LAG(wFR,3)

(9) pIT=.4366*wIT+.3122*LAG(wIT,1)+.1878*LAG(wIT,2)+.0634*LAG(wIT,3)

(10) wDE=.32305*pDE+.27435*LEAD(pDE,1)+.22565*LEAD(pDE,2)+.17695*LEAD(pDE,3)+

(11) wFR=.42835*pFR+.30945*LEAD(pFR,1)+.19055*LEAD(pFR,2)+.07165*LEAD(pFR,3)+

(12) wIT-.4366*pIT-.3122*LEAD(pIT,1)-.1878*LEAD(pIT,2)-.0634*LEAD(pIT,3)=

(13) vIT=.4366*(yIT-.4366*pIT-.3122*LEAD(pIT,1)-.1878*LEAD(pIT,2)-.0634*

(14) is=alphar*LAG(is,1)+alphapi*(.4248*(pDE-LAG(pDE,4))+.2922*

(15) rlDE=(1/8)*is+(1/8)*LEAD(is,1)+(1/8)*LEAD(is,2)+(1/8)*LEAD(is,3)+

(16) rlFR=(1/8)*is+(1/8)*LEAD(is,1)+(1/8)*LEAD(is,2)+(1/8)*LEAD(is,3)+

(17) rlIT=(1/8)*is+(1/8)*LEAD(is,1)+(1/8)*LEAD(is,2)+(1/8)*LEAD(is,3)+

(18) yDE_=0*one
```



```
(19) yFR_=0*one

(20) yIT_=0*one

(21) wDE_=0*one

(22) wFR_=0*one

(23) wIT_=0*one

(24) one=LAG(one,1)

Writing out Parser Information
Done.

Writing out Structural Coefficient Matrix
Done.

----------------------------------------------------------------------
Solving model   :  cw0r5
Model directory:  /windows/desktop/project
Parameter file :  setcw0.m
Expectations viewpoint:  period t.
----------------------------------------------------------------------

----------------------
Numerical Tolerances
----------------------
 condn      : 1e-008
 uprbnd - 1: 1e-006


----------------------
Parameter settings
----------------------
Name       Value
----------------------
alphar        0.25000
alphapi       0.25000
pistar        2.00000
```

alphay 0.50000

Number of exact shiftrights (nex): 156
Number of numeric shiftrights (nnum): 1
Number of large roots (lgrts): 35
Number of stability conditions (nex + nnum + lgrts) -
number required (neq*nlead) = 0
Dimension of state transition matrix (ia): 90
Aim return code: unique solution.

	Roots		Amplitude	Period
1	29.759		29.759	0
2	-16.371 +	16.123i	22.977	2.658
3	-16.371 -	16.123i	22.977	2.658
4	-3.4759 +	1.0861i	3.6416	2.2134
5	-3.4759 -	1.0861i	3.6416	2.2134
6	-2.2524 +	2.6144i	3.4509	2.7534
7	-2.2524 -	2.6144i	3.4509	2.7534
8	1.1292 +	3.2261i	3.4181	5.0913
9	1.1292 -	3.2261i	3.4181	5.0913
10	-1.0674 +	3.2393i	3.4106	3.326
11	-1.0674 -	3.2393i	3.4106	3.326
12	2.7448 +	1.3681i	3.0669	13.588
13	2.7448 -	1.3681i	3.0669	13.588
14	-1.9325 +	2.2778i	2.9871	2.7626
15	-1.9325 -	2.2778i	2.9871	2.7626
16	-2.6155		2.6155	2
17	-1.7978 +	1.8955i	2.6124	2.697
18	-1.7978 -	1.8955i	2.6124	2.697
19	-2.518 +	0.52392i	2.5719	2.1397
20	-2.518 -	0.52392i	2.5719	2.1397
21	0.27732 +	2.4145i	2.4303	4.3141
22	0.27732 -	2.4145i	2.4303	4.3141
23	-0.74374 +	2.2241i	2.3452	3.3183
24	-0.74374 -	2.2241i	2.3452	3.3183
25	-1.672 +	1.548i	2.2785	2.6238
26	-1.672 -	1.548i	2.2785	2.6238
27	1.7236 +	1.1336i	2.063	10.8
28	1.7236 -	1.1336i	2.063	10.8
29	0.30657 +	2.0389i	2.0618	4.4199
30	0.30657 -	2.0389i	2.0618	4.4199
31	1.0344 +	1.1757i	1.566	7.3983
32	1.0344 -	1.1757i	1.566	7.3983
33	1.3812		1.3812	0
34	0.9998 +	0.047931i	1.0009	131.16
35	0.9998 -	0.047931i	1.0009	131.16
36	1		1	0
37	1		1	0
38	0.97048		0.97048	0
39	0.95504		0.95504	0
40	0.93249		0.93249	0
41	0.89985		0.89985	0
42	0.33372 +	0.32402i	0.46514	8.153
43	0.33372 -	0.32402i	0.46514	8.153
44	0.34182 +	0.033262i	0.34344	64.773
45	0.34182 -	0.033262i	0.34344	64.773
46	-0.20866 +	0.2641i	0.33659	2.8057
47	-0.20866 -	0.2641i	0.33659	2.8057
48	0.28645		0.28645	0
49	-0.10977 +	0.23784i	0.26195	3.1366
50	-0.10977 -	0.23784i	0.26195	3.1366
51	-0.17197 +	0.19671i	0.26128	2.7447
52	-0.17197 -	0.19671i	0.26128	2.7447
53	-0.20471 +	0.12472i	0.23971	2.4218
54	-0.20471 -	0.12472i	0.23971	2.4218
55	-0.14908		0.14908	2
56	0.040303 +	0.017946i	0.044118	14.998
57	0.040303 -	0.017946i	0.044118	14.998
58	-0.040808 +	0.015716i	0.043729	2.265
59	-0.040808 -	0.015716i	0.043729	2.265
60	0.014262 +	0.041189i	0.043588	5.0775
61	0.014262 -	0.041189i	0.043588	5.0775
62	-0.018953 +	0.03913i	0.043479	3.1076
63	-0.018953 -	0.03913i	0.043479	3.1076
64	0.0040141		0.0040141	0
65	0.0020011 +	0.0034745i	0.0040095	5.994
66	0.0020011 -	0.0034745i	0.0040095	5.994
67	-0.0020037 +	0.0034644i	0.0040021	2.9989
68	-0.0020037 -	0.0034644i	0.0040021	2.9989

69	-0.0039992		0.0039992	2
70	0.0021188		0.0021188	0
71	0.0010441 +	0.0018333i	0.0021098	5.9663
72	0.0010441 -	0.0018333i	0.0021098	5.9663
73	-0.001056 +	0.0018033i	0.0020898	2.9912
74	-0.001056 -	0.0018033i	0.0020898	2.9912
75	-0.0020786		0.0020786	2
76	-0.00017744		0.00017744	2
77	9.0989e-005 +	9.21e-005i	0.00012947	7.9386
78	9.0989e-005 -	9.21e-005i	0.00012947	7.9386
79	-9.1821e-005 +	8.9053e-005i	0.00012791	2.6495
80	-9.1821e-005 -	8.9053e-005i	0.00012791	2.6495
81	9.6887e-015		9.6887e-015	0

```
Maximum absolute error: 7.9658e-010
The conditional forecast refers to (# of periods ahead): 2

>>
```


References

Anderson, Gary S., "A Parallel Programming Implementation of the Anderson-Moore Algorithm", *unpublished manuscript*, Federal Reserve Board, 1997(a)

Anderson, Gary S., "Continuous Time Application of the Anderson-Moore (AIM) Algorithm for Imposing the Saddle Point Property in Dynamic Models", *unpublished manuscript*, Federal Reserve Board, 1997(b)

Anderson, Gary S., "The Anderson-Moore Algorithm: A MATLAB Implementation", *unpublished manuscript*, Federal Reserve Board, June 1999

Anderson, Gary S., "A Reliable and Computationally Efficient Algorithm for Imposing the Saddle Point Property in Dynamic Model", *unpublished manuscript*, Federal Reserve Board, September 2000

Anderson, Gary S., and George Moore, "A Linear Algebraic Procedure for Solving Linear Perfect Foresight Models", *Economics Letters*, Vol. 17, No. 1, 1985

Binder, Michael, and Hashem M. Pesaran, "Multivariate Rational Expectations Models and Macroeconometric Modeling: A Review and Some New Results", in Hashem M. Pesaran and Mike Wickens, *Handbook of Applied Econometrics*, Blackwell Publishers, 1995

Blanchard, Olivier J., and Charles M. Kahn, "The Solution to Linear Difference Models under Rational Expectations", *Econometrica*, Vol. 48, No. 5, 1980

Brayton, F., E. Mauskopf, D. Reifschenider, and John C. Williams, "A Guide to FRB/US: A Macroeconometric Model of the United States", *FEDS Discussion Paper*, No. 42, 1996

Clarida, Richard, Jordi Galì, and Mark Gertler, "Monetary Policy Rules and Macroeconomic Stability: Evidence and Some Theory", *NBER Working Paper*, No. 6442, 1998

Coenen, Günter, and Volker Wieland, "A Small Estimated Euro Area Model with Rational Expectations and Nominal Rigidities", *ECB Working Paper*, No. 30, September 2000

Coenen, Günter, and Volker Wieland, "Inflation Dynamics and International Linkages: A Model of the United States, the Euro Area and Japan", presented at the *International Research Forum on Monetary Policy*, Frankfurt am Main, July 2002

Coenen, Günter, Andrew Levin, and Volker Wieland, "Data Uncertainty and the Role of Money as an Information Variable for Monetary Policy", *ECB Working Paper*, No. 84, November 2001

Dennis, Richard, "Optimal Policy in Rational Expectations Model: New Solution Algorithms", *Working Paper*, Federal Reserve Bank of San Francisco, No. 9, July 2001

Fagan, Gabriel, Jerome Henry, and Ricardo Mestre, "An Area-Wide Model (AWM) for the Euro Area", *ECB Working Paper*, No. 42, January 2001

Finan, Frederico S., and Robert Tetlow, "Optimal Control of Large, Forward-Looking Models. Efficient Solutions and Two Examples", *Finance and Economics Discussion Paper*, Federal Reserve Board, No. 51, October 1999

Fuhrer, Jeffrey C., "Monetary Policy Shifts and Long-term Interest Rates", *Quarterly Journal of Economics*, Vol. 111, No. 4, 1996

Fuhrer, Jeffrey C., "Inflation/Output Variance Trade-offs and Optimal Monetary Policy", *Journal of Money, Credit, and Banking*, Vol. 29, No. 2, 1997(a)

Fuhrer, Jeffrey C., "The (Un)Importance of Forward-Looking Behavior in Price Specifications", *Journal of Money, Credit, and Banking*, Vol. 29, No. 3, 1997(b)

Fuhrer, Jeffrey C., and Hoyt Bleakley, "Computationally Efficient Solution and Maximum Likelihood Estimation of Nonlinear Rational Expectations Model", *Working Paper*, Federal Reserve Bank of Boston, No. 2, August 1996

Fuhrer, Jeffrey C., and Brian Madigan, "Monetary Policy when Interest Rates are Bounded at Zero", *Review of Economics and Statistics*, Vol. 79, No. 4, 1997

Fuhrer, Jeffrey C., and G. R. Moore, "Inflation Persistence", *Quarterly Journal of Economics*, Vol. 110, 1995(a)

Fuhrer, Jeffrey C., and G. R. Moore, "Monetary Policy Trade-offs and the Correlation between Nominal Interest Rates and Real Output", *American Economic Review*, Vol. 85, No. 1, 1995(b)

King, Robert J., and Mark W. Watson, "System Reduction and Solution Algorithms for Singular Linear Difference Systems under Rational Expectations", *International Economic Review*, Vol. 39, No. 4, November 1998

Klein, Paul, "Using the Generalized Schur Decomposition to Solve a Multivariate Linear Rational Expectations Model", *Journal of Economic Dynamics and Control*, Vol. 24, No. 10, 2000

Levin, Andrew, Volker Wieland, and John C. Williams, "The Robustness of Simple Monetary Policy Rules under Model Uncertainty", in John B. Taylor (ed), *Monetary Policy Rules*, Chicago University Press, 1999

Levin, Andrew, Volker Wieland, and John C. Williams, "The Performance of Forecast-Based Monetary Policy Rules under Model Uncertainty", *unpublished manuscript*, Federal Reserve Board, April 2001

Orphanides, Athanasios, "Monetary Policy Evaluation with Noisy Information", *FEDS Working Paper*, No. 50, 1998

Orphanides, Athanasios, and Volker Wieland, "Price Stability and Monetary Policy Effectiveness when Nominal Interest Rates Are Bounded at Zero", *FEDS Working Paper*, No. 35, 1998

Orphanides, Athanasios, David Small, David Wilcox, and Volker Wieland, "A quantitative Exploration of the Opportunistic Approach to Disinflation", *FEDS Working Paper*, No. 36, 1997

Rudebusch, Glenn, "Assessing Nominal Income Rules for Monetary Policy with Model and Data Uncertainty", *Economic Journal*, Vol. 112, No. 79, 2002

Sims, Christopher, "Solving Linear Rational Expectations Models", *unpublished manuscript*, Yale University, May 1996

Svensson, Lars E. O., "Inflation Targeting as a Monetary Policy Rules", *NBER Working Paper*, No. 6790, November 1998

Söderlind, Paul, "Solution and Estimation of RE Macromodels with Optimal Policy", *European Economic Review*, Vol. 43, No. 2, 1999

Taylor, John B., "Aggregate Dynamics and Staggered Contracts", *Journal of Political Economy*, Vol. 88, No. 1, 1980

QUADERNI DEL DIPARTIMENTO DI ECONOMIA degli ultimi 5 anni

102 **Debora REVOLTELLA**, *Financing enterprises in the Czech Republic: the importance of firm-specific variables*, gennaio 1998.

103 **Cristiana PERONI**, *Modelli di previsione a breve termine dei tassi di cambio*, marzo 1998.

104 **Massimiliano BRATTI**, *L'evoluzione dei divari settoriali di valore aggiunto per addetto nei paesi OCSE*, marzo 1998.

105 **Tommaso LUZZATI**, *To what extent is the notion of efficiency relevant to Economics? Implications for Ecological Economics*, marzo 1998.

106 **Renato BALDUCCI**, *Concertazione tra le parti sociali e disoccupazione*, maggio 1998.

107 **Maura FRANCESE, Maria Teresa MONTEDURO**, *Does the functional form matter? A sensitivity analysis of female labour supply in the U.K and italy*, maggio 1998.

108 **Roberto ESPOSTI**, *Stochastic Technical Change and Procyclical TFP The Italian Agriculture Case*, ottobre 1998.

109 **Alessandro STERLACCHINI**,*Do innovative activities matter to small firms in non-R&D-intensive industries? An application to export performances*, novembre 1998.

110 **Stefano STAFFOLANI**, *Contratti di lavoro con informazione asimmetrica bilaterale: chi paga i bonus condizionali?*, dicembre 1998.

111 **Davide CASTELLANI, Antonello ZANFEI**, *Multinational experience and the creation of linkages with local firms. Evidence from the electronics industry*, dicembre 1998.

112 **Roberto ESPOSTI**, *Spillover tecnologici e origine della tecnologia agricola*, aprile 1999.

113 **Luca PAPI, Debora REVOLTELLA**, *Foreign Direct Investment in the Banking Sector: a Transitional Economy Perspective*, aprile 1999.

114 **Roberto ESPOSTI, Franco SOTTE**, *Territorial Heterogeneity and Institutional Structures in Shaping Rural Development Policies in Europe*, luglio 1999.

115 **Renato BALDUCCI**, *Crescita endogena e ciclo*, luglio 1999.

116 **Antonio G. CALAFATI**, *Evoluzione dei sistemi locali e conservazione nei Parchi naturali*, luglio 1999.

117 **Renato BALDUCCI, Stefano STAFFOLANI**, *Distribuzione e crescita in un modello di contrattazione con impegno endogeno*, agosto 1999.

118 **Marco CUCCULELLI**, *Competizione sui mercati internazionali e misure del potere di mercato. Il caso dell'industria italiana dell'elettrodomestico*, agosto 1999.

119 **Riccardo LUCCHETTI**, *Analytic Score for Multivariate GARCH Models*, ottobre 1999.

120 **Alberto BUCCI**, *Horizontal innovation, market power and growth*, ottobre 1999.

121 **Riccardo LUCCHETTI, Luca PAPI, Alberto ZAZZARO**, *Efficienza del sistema bancario e crescita economica nelle regioni italiane*, ottobre 1999.

122 **Francesco TROMBETTA,** *Quanto costa controllare la natura? Il caso Mississippi*, ottobre 1999.

123 **Massimo TAMBERI,** *Nel mosaico economico delle marche: origini e trasformazioni*, novembre 1999.

124 **Stefano SANTACROCE,** *Graduates in the Labour Market, Determinants of Employment Success*, dicembre 1999.

125 **Massimiliano BRATTI,** *A study of the differences across universities in students' degree performance: the role of conventional university inputs*, dicembre 1999.

126 **Davide BERLONI, Roberto ESPOSTI,** *Scelte residenziali e mercati locali del lavoro. Il caso delle marche*, dicembre 1999.

127 **Davide TICCHI,** *Investment and uncertainty with recursive preferences*, gennaio 2000.

128 **Fabio FIORILLO, Stefano STAFFOLANI,** *To redistribute or not? Unemployment benefit, workfare and citizen's income in a dual labour market*, marzo 2000.

129 **Davide IACOVONI, Alberto ZAZZARO,** *Legal System Efficiency, Information Production, and Technological Choice: A Banking Model*, aprile 2000.

130 **Riccardo MAZZONI,** *I fattori di competitività dei settori tradizionali italiani: sintesi di un dibattito*, aprile 2000.

131 **Antonio G. CALAFATI,** *How Do Collective Agents Think?*, aprile 2000

132 **Antonio G. CALAFATI,** *Albert O. Hirschman on Economic Evolution*, aprile 2000.

133 **Antonio G. CALAFATI,** *On Industrial Districts*, aprile 2000.

134 **Alberto BUCCI,** *On Scale Effects, Market Power and Growth when Human and Technological Capital are Complements*, maggio 2000.

135 **Luca PAPI, Alberto ZAZZARO,** *How Does the EU Agenda Influence Economies Outside the EU? The Case of Tunisia*, giugno 2000.

136 **Roberto ESPOSTI,** *Public R&D Design and Technological Spill-Ins. A Dynamic Model*, giugno 2000.

137 **Alessandro STERLACCHINI,** *L'accesso alle professioni regolamentate: un' analisi empirica sui laureati degli atenei marchigiani*, luglio 2000

138 **Alberto BUCCI, H. Cagri SAGLAM,** *Growth Maximizing Patent Lifetime*, luglio 2000.

139 **Riccardo MAZZONI,** *Alcuni vincoli del processo di accumulazione*, agosto 2000.

140 **Riccardo LUCCHETTI,** *Inconsistency Of Naive GMM Estimation For QR Models With Endogenous Regressors*, luglio 2000.

141 **Alberto BUCCI, Fabio FIORILLO, Stefano STAFFOLANI,** *Can Market Power influence Employment, Wage Inequality and Growth?*, ottobre 2000.

142 **Alessandro STERLACCHINI,** *The determinants of export performance: A firmlevel study in Italian Manufacturing*, ottobre 2000.

143 **Renato BALDUCCI, Stefano STAFFOLANI,** *Quota del lavoro e occupazione in presenza di contrattazione efficiente*, ottobre 2000.

144 **Giorgio BARBA NAVARETTI, Enrico SANTARELLI, Marco VIVARELLI,** *The Role of Subsidies in Promoting Italian Joint Ventures in Least Developed and Transition Economies*, dicembre 2000.

145 **Roberto ESPOSTI, Pierpaolo PIERANI,** *Building the Knowledge Stock: Lags, Depreciation and Uncertainty in Agricultural R&D*, gennaio 2001.

146 **Francesco TROMBETTA,** *Il sistema economico locale di Fabriano e le sue articolazioni funzionali*, febbraio 2001.

147 **Antonio CALAFATI, Francesca MAZZONI,** *Conservazione, sviluppo locale e politiche agricole nei parchi naturali*, marzo 2001.

148 **Maria Rosaria CARILLO, Alberto ZAZZARO,** *Innovazione, ricerca della rendita e prestigio sociale: verso una teoria dinamica delle professioni*, maggio 2001.

149 **Marco GALLEGATI, Mauro GALLEGATI,** *European Business Cycles: 1960-1998*, maggio 2001.

150 **Luca NUNZIATA, Stefano STAFFOLANI,** *On Short-term Contracts Regulations*, maggio 2001.

151 **Massimiliano BRATTI,** *Oltre la scuola dell'obbligo. Un'analisi empirica della decisione di proseguire nell'istruzione post-obbligo in Italia*, maggio 2001.

152 **Massimiliano BRATTI, Stefano STAFFOLANI,** *Performance accademica e scelta della facoltà universitaria: aspetti teorici e evidenza empirica*, giugno 2001.

153 **Fabio FIORILLO, Giulio PALOMBA,** *Un modello CGE per l'analisi del federalismo fiscale all'italiana*, giugno 2001.

154 **Massimiliano BRATTI,** *Labour Force Participation and Marital Fertility of Italian Women: The Role of Education*, settembre 2001.

155 **Riccardo LUCCHETTI, Alessandro STERLACCHINI,** *Factors Affecting the Adoption of ICTs Among SMEs: Evidence From an Italian Survey*, ottobre 2001.

156 **F. CHELLI, L. ROSTI,** *Youth Unemployment and Self-Employment in Italy*, novembre 2001.

157 **Alberto ZAZZARO,** *The Allocation of Entrepreneurial Talent under Imperfect Lending Decisions*, novembre 2001.

158 **Luca De BENEDICTIS, Massimo TAMBERI,** *A note on the Balassa Index of Revealed Comparative Advantage*, gennaio 2002.

159 **Enrico GUZZINI,** *The Liberal Paradox and Non-Welfarist Theories: To What Extent Is There a Compatibility?*, febbraio 2002.

160 **Luca De BENEDICTIS, Massimo TAMBERI,** *Il modello di specializzazione italiano: normalità e asimmetria*, febbraio 2002.

161 **Marco GALLEGATI,** *Financial Constraints and the Balance Sheet Channel: a Re-Interpretation*, febbraio 2002.

162 **Paolo ZAGAGLIA,** *On (Sub)Optimal Monetary Policy Rules under Untied Fiscal Hands*, marzo 2002.

163 **Alberto ZAZZARO,** *How Heterodox Is the Heterodoxy of the Monetary Circuit Theory? The Nature of Money and the Microeconomy of the Circuit*, aprile 2002.

164 **Massimo GIULIODORI,** *Monetary Policy Shocks and the Role of House Prices Across European Countries,* maggio 2002.

165 **Giuseppe RICCIARDO LAMONICA,** *La funzionalità nelle zone omogenee delle Marche,* maggio 2002.

166 **Roberto ESPOSTI, Alessandro SORRENTINO,** *Regolamentazione delle Innovazioni Biotecnologiche in Agricoltura e Accordi Multilaterali: Conflitti, Negoziazione e Innovazione Istituzionale,* luglio 2002.

167 **Maria Rosaria CARILLO, Alberto ZAZZARO,** *The Enigma of Medieval Craft Guilds: A Model of Social Inertia and Technological Change,* luglio 2002.

168 **Fabiano COMPAGNUCCI,** *Sviluppo senza crescita: il sistema locale del Casentino,* luglio 2002.

169 **Paolo ZAGAGLIA,** *Matlab Implementation of the AIM Algorithm: A Beginner's Guide,* luglio 2002.