January 2011

AIM_SOLVE

(version 1.1)

Overview

The EViews **aim_solve** add-in is a user-friendly interface to the Anderson-Moore (AMA) algorithm (also known as AIM) for transforming linear rational expectations models into a form that contains no future-dated variables. The add-in requires R and the AMA package.[1] The primary input to the add-in is an EViews model that must be linear in variables, contain at least one lead of a variable, and have a dynamic structure that satisifies a stability condition. The primary output of the add-in is an EViews model that contains only contemporaneous and lagged variables and thus can be simulated using simple techniques. The procedure also creates a text file (*aim_solve_text*) containing summary information.

Syntax

The procedure, which must be executed from a command line or in a program, has two alternative forms:

A. *input_model_name*.aim_solve(*options*)

B. *input_model_name*.aim_solve(*options*) *arg1 arg2*

EViews assumes that each endogenous variable in a model will appear as the first variable in one (and only one) of its equations. Form A, which uses the EViews model processor to identify the endogenous variables, requires that the input model satisfy this normalization condition. Exogenous variables are not permitted in form A. If the model contains shock variables, their equations (typically, *shockvar=0*) need to explicitly appear in the input model. To ensure that the contemporaneous effects of any shocks are correctly transmitted, the output model is written in structural form.

$$\mathbf{y}_t = \mathbf{D_0}\mathbf{y}_t + \mathbf{D_1}\mathbf{y}_{t-1} + \ldots, \tag{1}$$

[1] The Anderson-Moore algorithm is described in Anderson and Moore (1985) and Anderson(2008, 2010). The R version of AMA was written by Gary Anderson and Aneesh Raghunandan. See the "notes" section for instructions on how to install AMA.

In (1), $\mathbf{y}$ is the vector of model variables (including shocks) and each $\mathbf{D_i}$ is a matrix of structural coefficients.

It may be natural to express the equations of some models in a way that does not satisfy the normalization requirement. Form B is available for use in such cases, as an alternative to creating a normalized model by hand. In this variant of the proc, two string variables define the endogenous (*arg1*) and shock (*arg2*) variables. An option is available to instruct the procedure to add the shock equations, if they are not already in the input model. In the output model, the shock equations are omitted and the only contemporaneous variables that appear on the right hand side of equations are the shocks.

$$\mathbf{z}_t = \mathbf{B_1}\mathbf{z}_{t-1} + \ldots + \mathbf{C}\mathbf{x}_t. \tag{2}$$

Here, $\mathbf{z}$ is the vector of endogenous variables (excluding shocks), $\mathbf{x}$ is the vector of shocks, each $\mathbf{B_i}$ is a matrix of reduced-form coefficients, and $\mathbf{C}$ is a matrix of structural shock coefficients.

Equations 1-2 are matrix representations of the output model. In fact, the procedure creates an EViews model in which by default each equation contains only those variables whose coefficients are not zero. The design of simulations of the output model will depend on which version of the procedure is used. In form A, the introduction of perturbations may require add factors or the "exclusion" of one or more shock equations, whereas in form B the shock variables are exogenous and can be perturbed directly.

Options

	description	default
parse=*s/a*	Parsing method	A:*s*; B:*a*
addshks	Add shock equations (form B)	
modout=*text*	Name of output model	*aimmod*
coefpre=*text*	Coefficient prefix in output model	*c_*
mat=*no/yes/only*	Option for retaining AMA matrices in workspace	*no*
z=1/2	Option for how may variables to include in output model equations	1
debug	Do not delete temporary matrices	

The *aim_solve* proc passes the input model to the AMA package as a matrix of coefficients. The proc has two methods for parsing the input model to create this coefficient matrix: one based on static *simulations*, the other on *arithmetic* operations. The first method requires that the input model have a form that EViews can simulate; the second does not. Because form A of the procedure already requires that the input model meet the key normalization attribute of a valid EViews model, the simulation-based parser (**parse**=*s*) is the default in this case. On the other hand, because most models that qualify for form A of the command can also be successfully parsed using the other option, the best choice of parser may depend on relative speed, an aspect of parser performance that varies with the specific characteristics of each model.

Because the design of form B of the procedure permits models that do not satisfy the normalization condition, the arithmetic parser is (**parse**=*a*) the logical default in this case. This parser only requires that each equation be a legal arithmetic expression. When it is desired to use form B with an input model that meets the normalization criteria, the simulation-based parser may be used. The **addshks** option instructs version B of the procedure to add an equation of the form *shockvar=0* for each variable in *arg2*.

Under the default settings, the name of the output model is *aimmod* and the names of its coefficients are based on adding a *c_* prefix to the name of the endogenous variable associated with each equation. The **modout** and **coefpre** options may be used to select other model and coefficient names. The **mat** option controls the retention of the AMA structural (*scof*) and reduced-form (*bmat*) coefficient matrices in the workspace. The order in which variables/equations appear in these matrices is given by the variable/equation order contained in the string *varlist*. When **mat**=*only*, the matrices are saved in the workspace and the output EViews model is not constructed. The **z** option controls how many variables with zero coefficients remain in the output model. When **z**=*1* (the default), each equation contains only those variables that have non-zero coefficients in that equation. When **z**=*2*, each equation contains all variables that have non-zero coefficients in at least one equation. The latter setting reduces processing time but increases the time required to simulate the output model. When the **debug** option is specified, temporary working objects are not deleted and their names are reported in the text object **aim_solve_work**.

Examples[2]

```
testmod.aim_solve

testmod.aim_solve(parse=a,modout=mymod,coefpre=x_)

string evar = "z1 z2 z3 z4"
string svar = "e1 e2"
testmod.aim_solve(addshks,mat=yes) evar svar
```

In the first example, the procedure takes the model *testmod* and creates an output model named *aimmod* of the form given by equation 1. In the second example, the input model is parsed using the *arithmetic* approach, the output model is named *mymod*, and its coefficients have an $x\_$ prefix. In the third example, the string variables *evar* and *svar* contain the input model's endogenous and shock variables. A shock equation is added to the input model for each variable in *svar*. The procedure creates an output model named *aimmod* of the form given by equation 2. The AMA matrices *bmat* and *scof* and string *varlist* remain in the workspace.[3]

Notes

Consider a linear model with m lags and n leads of the vector variable $\mathbf{y}$,

$$\mathbf{H}_{-m}\mathbf{y}_{t-m} + \ldots + \mathbf{H}_0\mathbf{y}_t + \ldots + \mathbf{H}_n\mathbf{y}_{t+n} = 0 \tag{3}$$

Each $\mathbf{H}_i$ is a square coefficient matrix. For models that satisfy the Blanchard-Kahn conditions, AMA computes the coefficients of the structural and reduced-form solutions,

$$\begin{aligned} \mathbf{S}_m\mathbf{y}_{t-m} + \ldots + \mathbf{S}_0\mathbf{y}_t &= 0 & (4)\\ \mathbf{B}_m\mathbf{y}_{t-m} + \ldots + \mathbf{B}_1\mathbf{y}_{t-1} &= \mathbf{y}_t & (5) \end{aligned}$$

The reduced-form coefficients ($\mathbf{B}$) are related to the structural coefficients ($\mathbf{S}$) by the formula,

[2]For a more detailed presentation of how to use *aim_solve*, see the two EViews programs that are part of the add-in distribution.

[3]Equations 7 and 8 given the specific forms of *bmat* and *scof*.

$$\mathbf{B}_i = -\mathbf{S}_0^{-1}\mathbf{S}_i. \tag{6}$$

The aim_solve add-in has three steps. After determining the values of the maximum lag (m) and lead (n), either static simulations or aritmetic operations are executed to calculate the elements of each $\mathbf{H}_i$. The second step passes this matrix form of the input model to R, calls AMA to solve for the structural and reduced-form coefficient matrices,

$$\mathbf{scof} = [\mathbf{S}_m \dots \mathbf{S}_0] \tag{7}$$
$$\mathbf{bmat} = [\mathbf{B}_m \dots \mathbf{B}_1] \tag{8}$$

and moves the solution matrices back to the EViews workspace. The last step assembles the EViews equation strings that form the output model.

To install (or update) the AMA package, start R and enter:

```
options(repos=c(CRAN = "ftp://cran.r-project.org/pub/R"))
install.packages("AMA")
```

References

Anderson, G. and Moore, G. "A Linear Algebraic Procedure For Solving Linear Perfect Foresight Models." Economics Letters, 17, 1985.

Anderson, G. "Solving Linear Rational Expectations Models: A Horse Race." Computational Economics, 2008, vol. 31, issue 2, pp. 95-113.

Anderson, G. "A Reliable and Computationally Efficient Algorithm for Imposing the Saddle Point Property in Dynamic Models." Journal of Economic Dynamics and Control, 2010, vol 34, issue 3, pp. 472-489.