An Overview of the Sym Language and Conversion Program

August 2018

Summary:

Sym is a set-driven matrix language for expressing large-scale general equilibrium models in algebraic form. It is designed to be nearly as concise as standard matrix notation. However, it also imposes rigorous conformability rules on all expressions, allowing it to catch a broad range of potential errors that could otherwise arise in matrix-driven languages. As a result, subscripts are unnecessary in many expressions, improving the clarity of a model's code and facilitating documentation and maintenance. This document provides a brief overview of the language and how it is used.

An Introductory Example:

To illustrate the role of Sym's conformability rules, consider a simple example: calculating total expenditure by households of various types. Suppose there are two households, A and B, and two goods, X and Y. Let the prices of the goods be a two-element vector P; let the quantities purchased by the households be a 2x2 matrix Q with one row per household and one column per good; and let total household expenditure on the goods be two-element vector V.

In scalar notation, the elements of V would be computed as follows:

$$\begin{aligned} V_A &= Q_{A,X} P_X + Q_{A,Y} P_Y \\ V_B &= Q_{B,X} P_X + Q_{B,Y} P_Y \end{aligned} \tag{1}$$

This notation is unambiguous, which is an asset, but it is also verbose (making it cumbersome to read) and it is difficult to maintain when the dimensions of a model change (for example, adding

a household or a good). In contrast, the matrix notation version of the equation, if P and V were column vectors, could be computed as follows:

(2) V = Q*P

This form is clearly more compact, and it is also easier to maintain: adding households or goods would change the dimensions of the variables but would have no effect on the equation itself. However, it allows undesirable ambiguities to arise: under standard matrix conformability rules Q*P can be computed for any Q and P having dimensions with matching lengths regardless of whether the columns of Q and rows of P are both in the correct order, or even whether the columns of Q and rows of P range over the same domains. It is thus highly prone to programming errors; an example will be discussed further below.

Sym takes an intermediate approach that has most of the advantages of matrix notation in terms of clarity and maintainability but avoids its ambiguity. In Sym, the calculation of V would be expressed this way:

(3) V = sum(goods, Q*P)

Sym interprets the expression Q*P as an element-by-element operation that returns a 2x2 matrix in which each element of *Q* has been multiplied by an element of *P* determined by aligning the domains (sets over which subscripts range) of each variable:

(4) $$Q * P = \begin{bmatrix} Q_{A,X} * P_X & Q_{A,Y} * P_Y \\ Q_{B,X} * P_X & Q_{B,Y} * P_Y \end{bmatrix}$$

The sum is then taken over goods (that is, along the rows), which returns the two values, V_A and V_B, shown in the earlier scalar equation.

Sym matches elements of Q and P using the logical definitions of the subscripts involved (in this case, X and Y or A and B) rather than the order of elements within the vector or matrix.

Each element of the X column of Q is multiplied by the X element of P no matter how the elements of P and Q are ordered. As a result, Q*P and P*Q return identical 2x2 matrices, and there is no need for the equation to include explicit subscripts. Sym enforces conformability between Q and P by checking (in this case) that exactly one of the dimensions of Q is defined over the same set of subscripts as vector P. This condition is stronger than that imposed by standard linear algebra: not only must the lengths of the vectors be conformable, but their logical definitions must be as well. All operations are carried out based on the values of the subscripts (such as X and Y), and do not depend on the order of the elements within the set: the sets {X,Y} and {Y,X} are equivalent.

Extending the Example to Higher Dimensions

The logic is the same for arrays with more dimensions. If Q were defined over two households {A,B}, two goods {X,Y}, and two regions {1,2}, and P were defined over goods {X,Y} and regions {1,2}, the expression Q*P would produce a 2x2x2 array where each element of Q is multiplied by the corresponding element of P. The element of Q*P corresponding to household A's consumption of good X in region 1, for instance, would be $Q_{A,X,1}P_{X,1}$. Array *V*, expenditure by each household, would now be a four-element array with dimensions {A,B}x{1,2}: one total for each household in each region. However, despite the change in dimensionality, the equation for V would look exactly the same as before:

(5) V = sum(goods, Q*P)

In both cases, the economic logic would be the same: the equation would calculate total expenditure by household. As a result, Sym allows the size and scope of a model to be changed simply by revising the sets over which variables are defined. Many equations need no modifications at all.

Continuing the example, the total amount spent on each good by both households together, *Vg*, could be computed by summing over the set households, defined to be {A,B}:

(6) Vg = sum(households, Q*P)

In this case, Vg would have one element for each good in each region rather than one per household. Total expenditure, *Vt*, could be calculated via any of the following:

(7) Vt = sum(goods, sum(households, Q*P))

(8) Vt = sum(households, V)

(9) Vt = sum(goods, Vg)

In the single region case, *Vt* would be a scalar while in the multi-region case it would be a vector with one element per region. The actual Sym code declaring the variables and carrying out all of these calculations in the single-region case is shown in Figure 1.

Figure 1: Sym code for matrix multiplication

```
set households (a, b) ;
set goods (x, y) ;

variable p(goods)
variable q(households, goods) ;
variable v(households) ;
variable vg(goods) ;
variable vt ;

v = sum(goods, q*p) ;
vg = sum(households, q*p) ;
vt = sum(households, v);
```

A key computational problem that Sym is designed to solve is that standard matrix notation imposes such weak conformability rules that it is easy to write mathematically-legitimate but nonsensical equations that are difficult to detect and debug. Equation (2), for

example, is mathematically valid as long as the number of columns of Q matches the number of rows of P. Either of the following could be computed, although only the first actually makes economic sense:

(10) $$\begin{bmatrix} V_A \\ V_B \end{bmatrix} = \begin{bmatrix} Q_{A,X} & Q_{A,Y} \\ Q_{B,X} & Q_{B,Y} \end{bmatrix} \begin{bmatrix} P_X \\ P_Y \end{bmatrix}$$

(11) $$\begin{bmatrix} V_A \\ V_B \end{bmatrix} = \begin{bmatrix} Q_{A,X} & Q_{B,X} \\ Q_{A,Y} & Q_{B,Y} \end{bmatrix} \begin{bmatrix} P_X \\ P_Y \end{bmatrix}$$

Even though the dimensions of Q and P match in the second equation, the result is clearly not what was intended:

(12) $$\begin{aligned} V_A &= Q_{A,X} P_X + Q_{B,X} P_Y \\ V_B &= Q_{A,Y} P_X + Q_{B,X} P_Y \end{aligned}$$

The only difference between the two formulations is really the definition of Q itself, which is transposed in (11). Given that definition of Q, the second equation should have been written:

(13) $$V^T = P^T * Q$$

That is:

(14) $$\begin{bmatrix} V_A & V_B \end{bmatrix} = \begin{bmatrix} P_X & P_Y \end{bmatrix} \begin{bmatrix} Q_{A,X} & Q_{B,X} \\ Q_{A,Y} & Q_{B,Y} \end{bmatrix}$$

Sym's stronger conformability rules prevent errors of this kind. They also prevent errors that could arise when variables are defined over inconsistent sets of subscripts. For example, given the definitions of Q and P shown in Figure 1, the two are conformable in Q*P because Q has two columns subscripted by the elements of set "goods" and P has two elements subscripted by the same set. Had P been defined over a different 2-element set, the product would not have been conformable even though the number of columns of Q and rows of P would match. Sym generates compile-time error messages when conformability rules are violated. For example, the

code in Figure 2 would trigger an error message. In this case, P is defined over "factors" rather than "goods" so Q*P is no longer conformable even though it is still a 2x2 matrix multiplied by a 2-element vector. Note that Sym considers any text following two consecutive slashes to be a comment.

Figure 2: Sym code triggering a conformability error

```
set households (a, b) ;
set goods (x, y) ;
set factors (l, k) ;

variable p(factors)
variable q(households, goods) ;
variable v(households) ;

v = q*p ; // would generate an error
```

Because subscript domains are used in determining conformability, Sym is able to determine the economically correct interpretation of any binary operation without regard to the ordering of the variables. The result is that models can be expressed very compactly and many potential opportunities for coding errors are eliminated.