Fast Re-Calculation of the Covariance Matrix Implied by a Recursive Structural Equation Model

by

Thomas Richardson

January 1996

Report CMU-PHIL-67

Fast Re-Calculation of the Covariance Matrix Implied by a Recursive Structural Equation Model

Thomas Richardson*

December 20, 1995

The following note presents a method for quickly recalculating the implied covariance matrix (E(0))of a recursive linear structural equation model, when one parameter in the model is changed. Fast re-calculation is required in order to make practical the use of Gibbs sampling techniques with linear structural equation models.

Statement of the problem

The Gibbs sampling technique requires that for every sample 'draw[1] from the posterior distribution, for every parameter, it is necessary to evaluate the likelihood function of the model; sometimes several times.

The algorithm used by the Gibbs sampler is such that only one model parameter (0j) is changed at a time, this feature can be exploited since some parts of the implied covariance matrix will be unaffected by changes to certain parameters, and thus these elements of the covariance matrix do not need to be re-calculated. While it is true that changing some of the parameters in a model will result in a change in the value of all of the entries in the implied covariance matrix, for a recursive model this cannot be the case in general.

Moreover, even though it may be the case that changing certain parameters will result in changes to many entries in the covariance matrix, work can still be cut-down here by making use of the fact that parts of the expression for some implied covariance will be very similar to certain pieces of the expression of other implied covariances. Moreover, such similarities are reflected in the structure of the directed paths and treks between pairs of variables in the graph.

In this note we first give recursive relationships from which we will derive algebraic expressions for the covariances and variances. Then we give a data structures which can be used to encode these expressions. Finally we show how it is possible from this data structure to construct an ordered set of quantities which must be re-calculated when a given parameter is changed, in order to re-calculate the implied covariance matrix. Thus we can avoid having to carry out any tests, concerning what to recalculate, while the sampler is running; all tests are carried out beforehand. All that is calculated when a given parameter is

* **I thank Richard Scheines, Steven Klepper, Peter Spirtes, Clark Glymour and Chris Meek for helpful conversations.**

changed are exactly those elements of the implied covariance matrix which change. The recurrence relations ensure that there is little or no redundancy in recalculating these parts of the covariance matrix.

Representation of the Covariance Matrix

We begin by deriving algebraic expressions for the implied covariance matrix. The basic principle that we employ is to try, wherever possible, to represent covariances in terms of other covariances. In this way the amount of re-calculation will be minimized.

By employing a number of simple recurrence relations we can show that the *form* of the espressions that we will need to represent are in fact quite simple. We can represent any covariance or variance fact by an expression of the form:

$$\mathrm{Cov}(X,Y) = a_t \bullet [\,.\,] + a_2 \bullet [\,.\,] + \ldots + a_n \bullet [\,.\,] + V(e_x) H(e_x, Y)$$

Where the ai expressions are model parameters, while the place-holders [.] are filled by other covariance facts , and n $\lesssim$ r where r is the max number of parents possessed by any variable in the graph. H(ex,Y) is a term whose value is Cov(ex,Y)/V(ex) and which is also calculated recursively.

We now give the recursive procedure from which we will derive expressions for the covariances and variances. These expressions will then be encoded using the data structure described in the next section.

A couple of small points of notation:

First, we assume that the graph is represented in a form in which every non-error variable X has associated with it a unique error variable ex. All error variables are uncorrelated. (i.e. if in the Bollen-style diagram there would have been an arc between two error terms say ex and ey, showing that they would have been correlated, then instead we put in a (latent) non-error variable as a common cause of X and Y, and keep the errors (ex,ey,£T) uncorrelated.)

Second we denote the coefficient on an edge between X and Y as Edge(X,Y) [or Edge(Y,X)].

Third, we include ex as one of the parents of X.

Procedure

1. Extend the partial ordering induced on the variables by the graph to a total ordering *(>•)*, such that a variable always comes after its ancestors and before its descendnts i.e. if X is an ancestor of Y in the graph then X ^ Y.

2. For every pair of variables X,Y such that X $\leq$ Y calculate

$$H(e_x,Y) := \frac{Cov(e_x,Y)}{V(e_x)}$$

Start with the variables which are last in the tl) ordering (i.e. those which come at the 'bottom' of the graph in the sense that they have no descendants). Then make use of the recurrence relation:

$$H(e_x,X) = l$$

$$H(e_x>Y)= \sum_{Z€Orildreo(X)} £Edge(X,Z)\text{-}H(e_Z,Y) \quad forX*Y$$

to form an expression for H(ex,Y) with X $\leq$ Y, by moving 'back' through the ordering i.e. only calculate an expression for H(ex,Y) if for all V >- X and W ^ Y [reverse lexicographic order] we have already calculated expressions for H(ev,W).Since children always come after their parents in the ordering we already know H(ez,Y) when calculating H(ex,Y). In this way we can express every term H(ex,Y) as a sum of products of two other quantities one of which we know directly [Edge(X,Z)], the other [H(ez,Y)] we have already formed an expression for. Note that H(£x,Y)*O if and only if Y is a descendant of X, hence no calculation is required to evaluate H(£x,Y) for Y-< X, since it is zero.

3. For any pair of variables X,Y, Cov(X,Y) can be expressed as:

$$Cov(X,Y) = \sum_{WeParents(X)} Edge(W,X).Cov(W,Y)$$

Where £x eParents(X). Distinguishing between ex and the other parents of X, then applying the definition of H(ex,Y), we arrive at the following expression, which we shall use for representing the covariance:

$$Cov(X,Y) = V(e_x) \bullet H(e_x,Y) + \sum_{W€Parents(X)\backslash\{£_x\}} Edge(W,X) \bullet Cov(W,Y)$$

If we now apply the above expression, starting with variables which come first in the ordering, (i.e. those that come at the top of the graph, in the sense that they have no ancestors), and such that X ^ Y w e can then proceed to express Cov(X,Y) again as a sum of products of two terms, one of which [Edge(W,X) or V(ex)] we know directly since they are model parameters the other of which [Cov(W,Y) or $H(e_x,Y)$] we have derived an expression for already, either in step 2 [H(ex,Y)] or in step 3 [Cov(W,Y)]. Again we proceed through the variables in reverse lexicographic order, though (in contrast to the step 2) we begin with the first elements in the -< ordering i.e. we calculate Cov(X,Y) only if we have already calculated Cov(V,W) for all V -< X, W r$\leq$ Y.

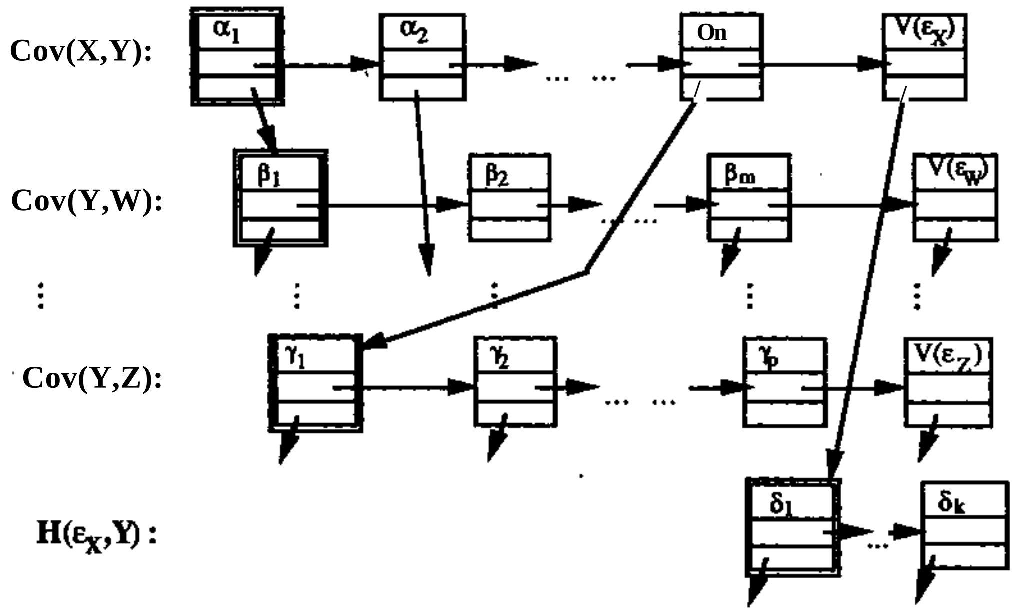
[This step also calculates variances since Cov(X,X)=V(X).]

It should be pointed out here, that we calculate expressions for covariances involving latent variables even though these are not required for the Gibbs procedure; the reason that we do this, is because these covariances are (in general) useful as sub-components of covariances between measured variables.

Data Structures

We now outline the data structures that we will make use of in order to encode the algebraic expressions for the covariance information. We first re-emphasize the point that we made earlier, that the Covariances could be expressed as a linear combination of other covariances, together with another term (H(£x,Y)), with the linear coefficients being model parameters. Thus we represent the expression for the covariance as a double linked list, one direction of linkage for the summation, the other for the recursion. The following example demonstrates this:

$$\mathrm{Cov}(X,Y) = \alpha_1 \cdot \mathrm{Cov}(Y, W) + a_4\mathrm{Cov}(Y,V) + \ldots -KX_A\mathrm{Cov}(Y,Z) + V(e_x > H(e_X Y)$$



(Arrows which do not point to anything are pointers to records which are not included in the diagram for reasons of space. In the actual implementation which we give below, the records have a number of additional fields, not shown in this diagram.)

We will refer to the lists which traverse the page horizontally as being 'sum lists[1]; it will be important later to distinguish the records which occur at the head of sum lists; in this example these are the records containing oci,pi, yi, 81, indicated by the double lined box.

The records in the data structure will take the following form:

Coefficient:	**Param #**
Next Term in Sum:	**Pointer**
Cov or H term:	**Pointer**
Value:	**Real**
Params occurring:	**Set**

Head Record Form

Coefficient:	**Param #**
Next Term in Sum:	**Pointer**
Cov or H term:	**Pointer**

Non-Head Record Form

Explanation of Fields:

Coefficient denotes records the parameter number of the coefficient (e.g. α_1,α_2 etc., in the above example).

Cov or H term is a pointer to the head of a linked list which gives the sum expressing the Cov(,) or H(,) to be multiplied by the Coefficient given by "Param #". When this pointer is empty, the coefficient is multiplied by 1.

Next Term in Sum is a pointer to the record giving the next term in the sum.When this pointer is empty, the term is the last in the sum.

Value is a field only in the head-records of a sum list. It contains the sum of the product of the coefficient and the value of the Cov or H term for each record in the list.

Params occurring is again a field only in the head-recordof a 'sum' list. It is used to record the set of model parameters numbers for the parameters that occur within a given expression at all levels of recursion e.g. for the record in which "β_1" occurs in the above example, *Params occuring* would be a set of model parameter numbers including the no.s for $\beta_1,\ldots\beta_m,V(\varepsilon_W)$, together with those for any parameters occuring in the linked lists pointed to by the records in which $\beta_1,\ldots\beta_m,V(\varepsilon_W)$ occur. This field is not used while the sampler is running; it is only used initially in order to work out which quantities must be recalculated when a given parameter is changed.

When implementing constructing this data structure, using the recurrence relations introduced in the last section, it is necessary to check whether a term is identically zero, e.g. $H(\varepsilon_X,Y)$, where Y is not a descendant of X, in which case the record can be omitted. Likewise there is no need to point to terms which are identically 1 e.g. $H(\varepsilon_X,X)$.(This should not present any great problems.)

Recalculation Lists

We now show how lists of which quantities must be re-calculated when a given parameter changes, can be inferred from the data structure that we have set up in the previous section. We first fill in the *Params Occuring* field in the head-records in the data structure. This can be achieved quite simply as follows:

1. For each of the sum lists corresponding to a (non-zero) H(ε_X,Y) do the following:

```
Begin with **CurrentRecord** as the Head record.
Store the location of the Head record.
Let **ParamsOcc** = { } (The empty set.)
(*) Let **ParamsOcc** = **ParamsOcc** $\cup$ {The no. in the *Coefficient* field of **CurrentRecord**} $\cup$
    The set in the *Params Occurring* Field of the Head-Record pointed to by the *Cov or H term* field
    of the **CurrentRecord**
If the *Next Term* field in **CurrentRecord** is non-null then let **CurrentRecord** = *Next Term.*, and
repeat the Step (*).
Else set **Current Record** = Head Record, and set *Params Occurring* = **ParamsOcc.**
```

It is important that the H(ε_X,Y) sum-lists are considered in the same (reverse-lexicographic) order that they were constructed. In this way the *ParamsOcc* field of any head-record that is consulted in Step (*) will already have been filled in.

2. For each of the sum lists corresponding to a (non-zero) Cov(X,Y) carry out exactly the same procedure as in **1**, once again considering the covariances in the same (reverse-lexicographic) order in which they were originally constructed.

After this procedure has been carried out the *Params Occurring* field in every head-record will be filled in. From here it is relatively simple to construct lists of quantities which must be recalculated whenevery a particular parameter is changed:

3. Construct a (linked) list of pointers to head-records, which we will call the re-calculation list, for each of the model parameters as follows:

Consider each of the non-zero H(ε ,) lists in turn, once again respecting the reverse lexicographic order as before. Now if a model parameter (e.g. α) is listed in the *Params Occurring* field of the head-record of a sum list (e.g. the list for H(ε_X,Y)) then append a pointer to this head-record to the re-calculation list for α.

Carry out the same procedure with the non-zero Cov(X,Y) lists, again respecting the reverse lexicographic order (appending pointers to the head-records of a sum list to the re-calculation list for α).

Use with the Gibbs Sampler

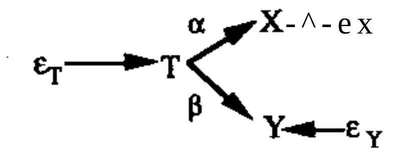
The re-calculation list is exactly what is required in order to efficiently 'update' the implied covariance matrix after changing a single model parameter (e.g. α). To update the implied covariance matrx we simply take each of the sum-lists referred to in the re-calculation list for α, in order. Taking each list in turn we carry out the sum as follows:

```
Begin with **CurrentRecord** as the Head record.
Store the location of the Head record.
Let **Val** = 0
(*) Let **Val** = **Val** + (The value of the parameter whose number is *Coefficient* in the **Current
Record**) $\times$ (The number in the *Value* field of the Head-Record pointed to by the *Cov or H term* field
of the **CurrentRecord** or 1 if the *Cov or H term* pointer of the **CurrentRecord** is null).
If the *Next Term* field in **CurrentRecord** is non-null then let **CurrentRecord** = *Next Term.*, and
repeat the Step (*).
Else set **Current Record** = Head Record, and set *Value* = **Val.**
```

The construction of the re-calculation lists ensures that no *Value* field is updated unless all those that it refers to have already been updated.
To extract the implied covariance matrix, all that is required is to look up the *Value* fields in the corresponding head-record of the list for that term.

Worked Example

In order to clarify the method, we go through the following worked example:



Model Parameters: V(ε_T),V(ε_X),V(ε_Y),α,β

1. Ordering: T ≺ X ≺ Y

2. Derivation of expressions for H(e,) terms, reverse lexicographic ordering, starting at end of -< ordering.

$H(e_Y,Y)$=l. $H(e_x,Y)$=O. $H(e_T,Y)$=Edge(T,Y·)$H(e_Y,Y)$=PH(e_Y,Y)=p.

$H(e_{XV}X)$=l• $H(e_T,X)$=Edge(T,X·)$H(e_x,X)$=a-$H(e_x,X)$=a

$H(e_T,T)$=l.

3. Derivation of expr. for Cov(,) terms, reverse lexicographic ordering, starting at beginning of -< ordering

Cov(T,T)=V(e_T)-H(e_T,T) = V(e_T) Cov(T,X)=V(e_T)H(e_T,X) Cov(T,Y)=V(e_T>H(e_T,Y)

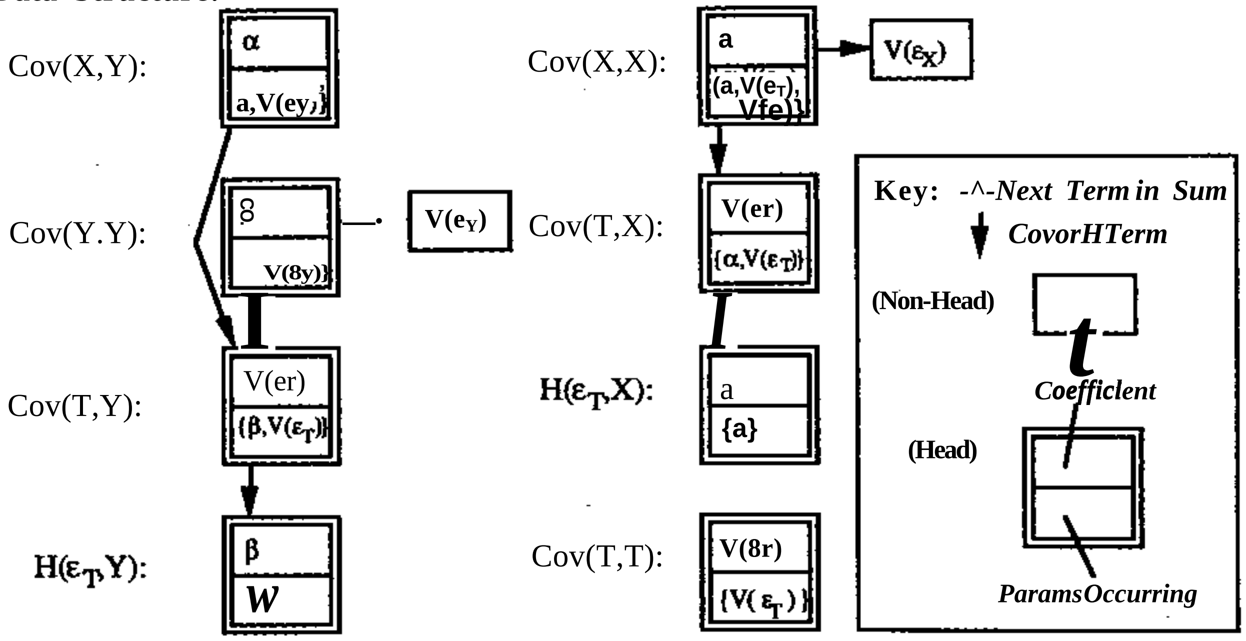
Cov(X,X)=V(e_x)-H(e_x,X)+Edge(X,T>Cov(T,X)

Cov(X,Y)=V(e_x)-H(e_x,Y)+Edge(X,T)-Cov(T,Y)= a-Cov(T,Y) (Since H(e_x,Y)=0.)

Cov(Y,Y)=V(e_Y)-H(e_Y,Y)+Edge(Y,T)-Cov(T,Y)= V(e_Y) + |3Cov(T,Y)

(We include the 'unsimplified' terms in order to illustrate the recurrence formulae.)

Data Structure:



Re-Calculation Lists

$V(e_T)$: Cov(T,T), Cov(T,X), Cov(T,Y), Cov(X,X), Cov(X,Y), Cov(Y,Y)

$V(e_x)$: Cov(X,X)

$V(e_Y)$: Cov(Y,Y)

a: $H(e_T,X)$, Cov(T,X), Cov(X,X), Cov(X,Y)

P: $H(e_T,Y)$, Cov(T,Y), Cov(X,Y), Cov(Y,Y)

[Note the re-calculation lists are ordered, so that if the given parameter is changed, then summing each of the sum-lists in turn will re-calculate the implied covariance matrix; if a sum list has another sum list as a term, then it is re-calculate first] <End of Example>

Complexity Calculations

In a worst case, in which the re-calculation list for a given parameter included every H-term and every Covariance fact, the total number of elements in the re-calculation list would be $O(n^2)$ where n is the number of variables (measured or unmeasured) occurring in the graph:

The numer of covariance/variance facts is: \ n(n +1)

The number of H(,) terms is: \ n(n -1)

Hence the maximum number of terms occurring in any re-calculation list is: n^2

The maximum number of calculations involved in the calculation of any given term is: 2r, where r = Max(Max no. of parents of any vertex* 1, Max no. of children of any vertex), since each term is expressed as a sum of at most r terms each of which is a product of 2 other terms, hence r multiplications, r additions, giving 2r operations in total for each H(,) or Covariance/Variance term.

Hence in the worst case, in which the re-calculation list for a given parameter includes every H(,) term and every Covariance/ Variance fact the Maximum number of calculations performed will be $2\text{-}r\text{-}n^2 < 2n^3$, (since a variable can have at most n-1 parents or children).

Of course in almost all cases the re-calculation list will have many fewer than n^2 many terms, it is for this reason that we carefully calculate these lists. However, even in a worst case this bound is

Conclusion:

The method advanced here is for acyclic graphs. It is not immediately clear how it might be generalized to cyclic models; it will be possible for both $H(e_x,Y)$ and $H(e_Y>X)$ to be non-zero, whereas this is not the case now (this is part of the reason the trek rule doesn't work with cyclic graphs). One method might be to convert the model to an (unfaithful) acyclic model in the manner outlined by Heise. Any method which uses the trek rule will fail to generalize; work is ongoing on finding a cyclic analog to the trek rule.

Bibliography

Scheines, R., Hoijtink, H., Boomsma, A., (1995) *Bayesian Estimation and Testing of Structural Equation Models* . CMU Philosophy, Methodology and Logic Technical Report CMU-PHIL-66

Scheines, R., Spirtes, P., Glymour, C. & Meek, C. (1994) *Tetrad II: Tools for Causal* ***Modeling. User's manual.*** **Hillsdale, NJ: Erlbaum.**

Spirtes, P., Glymour, C, & Scheines, R. (1993). Causation, prediction, and search. Springer-Verlag Lecture Notes in Statistics, 81. Springer-Verlag.