Arrays Plus Data Step Plus Cramer's Rule = Fast Rolling Regressions
Mark Keintz, Wharton Research Data Services, University of Pennsylvania

ABSTRACT
While the recent introduction of PROC FCMP has provided an effective way to improve the speed of SAS® in performing rolling window regressions, a simple data step application of Cramer's Rule for inverting matrices can be even faster. This paper describes how to generate rolling window regressions in a single data step using Cramer's Rule.

INTRODUCTION
Rolling windows regression, (more generally all rolling window analysis) is an analysis technique that lets the analyst explore patterns and relationships that change over time. However, instead of estimating a single model of a complete time series in which specific time periods might be modeled as predictor variables, the rolling window technique generates a model estimate for each window of a given size in the series. For instance a 5 year monthly data series would contain 49 windows of 12-months length.

The resulting primary SAS® programming issue is how to form, (or better yet, how to avoid forming), intermediate data sets that have each time point (DATE in the examples below) in the series available for analysis multiple times—once for each window containing that DATE. This is an issue whose significance becomes relevant (if not dominant) only as the size of the data series becomes large. In our case, we have 29,435 series with an average of 2,810 daily stock prices and returns. Analyzing rolling windows of 120 days length would generate almost 10 billion rows from this data set.

A previous paper by this author ("Rolling Regressions with PROC FCMP and PROC REG", see references) first showed "rolling data series" techniques which generated such a data set of complete time series data for subsequent submission to a PROC REG. It then demonstrated a faster technique using PROC FCMP to generate rolling sum-of-squares-and-cross-products (SSCP) for submission to PROC REG instead. This improved the performance results, primary because the rolling SSCP data set was much smaller than the rolling data series. However, it is possible to generate further performance improvements by using basic data step programming to not only generate the rolling SSCP, but also to invert the SSCP matrix in order to estimate regression parameters.

OVERVIEW OF THE “ROLLING DATA SERIES” APPROACH
A dataset of time series for 10,000 entities, each with, say, 5,000 daily records (e.g. a bit less than 20 years of stock market data) has about 50 million records, sorted by ID and DATE, might look like Table 1:

<table>
<thead>
<tr>
<th>ID</th>
<th>DATE</th>
<th>Y</th>
<th>X1</th>
<th>X2</th>
<th>X3</th>
<th>X4</th>
</tr>
</thead>
<tbody>
<tr>
<td>10107</td>
<td>1990/01/02</td>
<td>0.0198</td>
<td>0.0144</td>
<td>-0.0068</td>
<td>-0.0004</td>
<td>-0.0107</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>10107</td>
<td>2009/12/31</td>
<td>0.0071</td>
<td>0.0220</td>
<td>-0.0024</td>
<td>0.0164</td>
<td>0.0126</td>
</tr>
<tr>
<td>11081</td>
<td>1990/01/02</td>
<td>0.0225</td>
<td>0.0144</td>
<td>-0.0068</td>
<td>-0.0004</td>
<td>-0.0107</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>11081</td>
<td>2009/12/31</td>
<td>0.0236</td>
<td>0.0220</td>
<td>-0.0024</td>
<td>0.0164</td>
<td>0.0126</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>12490</td>
<td>2011/12/31</td>
<td>0.0113</td>
<td>0.0220</td>
<td>-0.0024</td>
<td>0.0164</td>
<td>0.0126</td>
</tr>
</tbody>
</table>
The simplest way to submit this data to rolling window regression is to produce an intermediate data set with one additional variable, call it WDATE (last date in the window). For windows of size 90, each WDATE would have 90 of the original records, (records 1-90 in window 1, 2-91 in window 2, etc.). Since almost every original record would appear in 90 windows, this new dataset would have nearly 4.5 billion records (90*50 million), and would likely look like Table 2 (sorted by ID WDATE, but not necessarily DATE). For techniques in constructing RWIND, see the “Brute Force” and “Holding and Writing” sections in KEINTZ [2012].

<table>
<thead>
<tr>
<th>ID</th>
<th>WDATE</th>
<th>(DATE)</th>
<th>Y</th>
<th>X1</th>
<th>X2</th>
<th>X3</th>
<th>X4</th>
</tr>
</thead>
<tbody>
<tr>
<td>10107</td>
<td>1990/05/09</td>
<td>(1990/01/02)</td>
<td>0.0198</td>
<td>0.0144</td>
<td>-0.0068</td>
<td>-0.0004</td>
<td>-0.0107</td>
</tr>
<tr>
<td>10107</td>
<td>1990/05/09</td>
<td>(1990/01/03)</td>
<td>0.0054</td>
<td>-0.0006</td>
<td>0.0072</td>
<td>-0.0029</td>
<td>-0.0034</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10107</td>
<td>1990/05/09</td>
<td>(1990/05/09)</td>
<td>0.0087</td>
<td>0.0080</td>
<td>-0.0071</td>
<td>-0.0028</td>
<td>0.0012</td>
</tr>
<tr>
<td>10107</td>
<td>1990/05/10</td>
<td>(1990/01/03)</td>
<td>0.0054</td>
<td>-0.0006</td>
<td>0.0072</td>
<td>-0.0029</td>
<td>-0.0034</td>
</tr>
<tr>
<td>10107</td>
<td>1990/05/10</td>
<td>(1990/01/04)</td>
<td>0.0292</td>
<td>-0.0071</td>
<td>0.0042</td>
<td>-0.0024</td>
<td>-0.0036</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10107</td>
<td>1990/05/10</td>
<td>(1990/05/10)</td>
<td>-0.0136</td>
<td>0.0009</td>
<td>-0.0030</td>
<td>0.0051</td>
<td>0.0010</td>
</tr>
<tr>
<td>10107</td>
<td>2009/12/31</td>
<td>(2009/08/25)</td>
<td>-0.0147</td>
<td>-0.0062</td>
<td>-0.0106</td>
<td>0.0022</td>
<td>-0.0050</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11081</td>
<td>1990/05/09</td>
<td>(1990/01/02)</td>
<td>0.0225</td>
<td>0.0144</td>
<td>-0.0068</td>
<td>-0.0004</td>
<td>-0.0107</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12490</td>
<td>2009/12/31</td>
<td>(2009/12/31)</td>
<td>0.0113</td>
<td>0.0220</td>
<td>-0.0024</td>
<td>0.0164</td>
<td>0.0126</td>
</tr>
</tbody>
</table>

At this point, actually running the rolling windows regression is trivial (per Figure 1 below):

```plaintext
PROC REG data=rwind;
    by id wdate;
    model y=x1 x2 x3 x4;
quit;
```

Figure 1: Regression on Rolling Data Series
THE ROLLING SSCP APPROACH

The performance challenge, of course, arises from the large size of RWIND, which is directly proportional to the window size (a 120 day window would result in 6 billion records). However, SAS has provided a way around this problem – by allowing PROC REG to read a special data set type: SSCP (Sum of Squares and Cross Products), which requires only NV+2 rows per window (NV is the number of variables in the regression model), regardless of the window size. Table 3 below (data set RSSCP) is all that is needed by PROC REG for a regression of Y on X1 through X4 for each ID/WDATE combination.

<table>
<thead>
<tr>
<th>ID</th>
<th>WDATE</th>
<th><em>TYPE</em></th>
<th><em>NAME</em></th>
<th>Intercept</th>
<th>X1</th>
<th>X2</th>
<th>X3</th>
<th>X4</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>10107</td>
<td>19900509</td>
<td>SSCP</td>
<td>Intercept</td>
<td>90</td>
<td>-0.0532</td>
<td>-0.1250</td>
<td>-0.0404</td>
<td>0.0286</td>
<td>0.5409</td>
</tr>
<tr>
<td>10107</td>
<td>19900509</td>
<td>SSCP</td>
<td>X1</td>
<td>-0.0532</td>
<td>0.0049</td>
<td>-0.0020</td>
<td>-0.0010</td>
<td>0.0002</td>
<td>0.0022</td>
</tr>
<tr>
<td>10107</td>
<td>19900509</td>
<td>SSCP</td>
<td>X2</td>
<td>-0.1250</td>
<td>-0.0020</td>
<td>0.0021</td>
<td>0.0002</td>
<td>0.0001</td>
<td>0.0018</td>
</tr>
<tr>
<td>10107</td>
<td>19900509</td>
<td>SSCP</td>
<td>X3</td>
<td>-0.0404</td>
<td>-0.0010</td>
<td>0.0002</td>
<td>0.0007</td>
<td>-0.0002</td>
<td>-0.0133</td>
</tr>
<tr>
<td>10107</td>
<td>19900509</td>
<td>SSCP</td>
<td>X4</td>
<td>0.0286</td>
<td>0.0002</td>
<td>0.0001</td>
<td>-0.0002</td>
<td>0.0009</td>
<td>-0.0010</td>
</tr>
<tr>
<td>10107</td>
<td>19900509</td>
<td>SSCP</td>
<td>Y</td>
<td>0.5409</td>
<td>0.0022</td>
<td>0.0018</td>
<td>-0.0133</td>
<td>-0.0010</td>
<td>0.1579</td>
</tr>
<tr>
<td>10107</td>
<td>19900509</td>
<td>SSCP</td>
<td>N</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>90</td>
</tr>
</tbody>
</table>

The Keintz [2012] paper provides a more detailed description of type SSCP data sets, as well as a way to create one with PROC EXPAND (from the SAS/ETS product). The required PROC REG syntax is still quite simple:

```
PROC REG data=rsscp (type=SSCP);
  by id wdate;
  model y=x1 x2 x3 x4;
quit;
```

MAKING ROLLING SSCP’S IN A DATA STEP

In lieu of PROC EXPAND, there are ways to create rolling SSCP’s within a DATA step. One, using the (relatively) recently added PROC FCMP was also demonstrated in the Keintz [2012] document. This had the benefit of being available in the Base SAS product, and also supported creation of a data set view (PROC EXPAND only outputs data set files) for input to PROC REG – reducing disk input/output usage. Its essential elements are repeated in Figure 3:
**Figure 3:**
Calling an FCMP-defined subroutine to make Rolling SSCP’s

```
DATA rolling_sscp / view=rolling_sscp;

** Allow for up to 10,000 daily values of Y, X1-X4 **;
array hist{10000,5} _temporary_; ** Send to FCMP **;
array rssc{10000,5,5} _temporary_; ** Receive from FCMP **;
array rsum{10000,5} _temporary_; ** Receive from FCMP **;

** Copy complete data series for an ID to _DATA matrix **;
  do nr=1 by 1 until (last.id);
    set ORIG;
    by id;
    do v=1 to 5; hist{nr,v}=var{v}; end;
  end;

** Ask FCMP subroutine for rolling SSCP and SUM **;
  if nr>=90 then call rssc(nr,90,2,hist,rssc,rsum);

  ... Additional code here to shape data into TYPE=SSCP form ... 
run;

PROC REG data=rolling_sscp ... ;
```

The program above (taken from Keintz [2012]) passes the array `hist` (complete history of a single ID) to an FCMP-defined subroutine named RSCCP, which then returns the rolling sscp values in array `rssc` (and rolling sums in `rsum`). While the entire program is not repeated here, the RSCCP subroutine definition is reproduced in Appendix 3.

**USING ARRAYS TO CONSTRUCT ROLLING SSCP**

The savings in time using this technique were significant, but only for larger windows (over 90 days in our tests). It seemed that the overhead of passing large arrays to and from the RSCCP subroutine reduced the performance benefits. The size of these arrays had to be made sufficiently large to accommodate (1) the longest possible history for any ID, and (2) the desired number of regression variables. I.e the longer the history, or bigger the model, the slower the task.

The script below demonstrates the construction of rolling SSCP’s for regression of Y on X1-X4 (plus a constant), without maintaining large arrays. The largest array required is no longer than the desired window size (90 in this case).
data results;
set orig;
by id;

Declare a 5×5 array to hold a single rolling SSCP. Since each SSCP data is based on multiple (90) successive records, the elements of the SSCP matrix are RETAINed. Also, to avoid extra calculations for this symmetric matrix, the elements below the diagonal are occupied by a dummy variable (xxx), used as a placeholder only. In addition an array is required to hold (and retain) sums of each variable:

array sumsqcp {5,5} _YY _YX1 _YX2 _YX3 _YX4
    XXX _X1X1 _X1X2 _X1X3 _X1X4
    XXX _X2X2 _X2X3 _X2X4
    XXX _X3X3 _X3X4
    XXX _X4X4 ;

array sumvars {5} _SY _S1 _S2 _S3 _S4 ;

retain _YY--_X4X4 _SY--_S4;

Because updating a rolling 90-day window requires removing the 90th preceding record when adding a new record, an array (HSTVARS) is needed to hold the most recent window (90 days) of historic values for Y and X1-X4. In addition, array HSTSQCP is needed to hold the corresponding historic values of their squares and cross-products. Since they are declared as _TEMPORARY_, their contents are automatically retained. Finally, variable _N is set up to track record sequence within each ID group:

array hstvars {90,5} _temporary_ (90*5*0);
array hstsqcp {90,5,5} _temporary_ (90*5*5*0);

retain _N; /* Counter within each entity */

At the beginning of every id, all the arrays above are initialized:

if first.id then do;
   _N=0;
   do r=1 to 5;
      sumvars{r}=0;
      do c=1 to 5; sumsqcp{r,c}=0; end;
      do h= 1 to 90;
         hstvars{h,r}=0;
         do c=1 to 5; hstsqcp{h,r,c}=0; end;
      end;
   end;
end;

In rolling windows, for each incoming observation, the oldest one must be replaced in the HST arrays and subtracted from the rolling sum arrays. Variable _H in the script below identifies what row in the HST arrays will be replaced. For example, observation 91 (_N=91) replaces observation 1, and will be placed in row 1 (_H=1). So will observations 181, 271, etc. Similarly observations 2, 92, 182, etc. will use row 2. The row is calculated by the MOD (for modulo) function, which produces the remainder from dividing the first argument by the second. The
HST arrays are updated only after the corresponding SUM arrays, (because the “exiting” row _H data must be subtracted before it is replaced by the new data).

\[ _N = _N + 1; \]
\[ _H = \text{mod}(_N - 1, 90) + 1; \]
do I=1 to 5;
  \text{sumvars}\{I\} = \text{sumvars}\{I\} - \text{hstvars}\{H,I\} + \text{vars}\{I\};
  \text{hstvars}\{H,I\} = \text{vars}\{I\};
do J=1 to 5;
  \text{xxx} = \text{vars}\{I\} * \text{vars}\{J\};
  \text{sumsqcp}\{I,J\} = \text{sumsqcp}\{I,J\} - \text{hstsqcp}\{H,I,J\} + \text{xxx};
  \text{hstsqcp}\{H,I,J\} = \text{xxx};
end;
end;
if _n < 90 then delete; /* Wait for a complete window */

APPLYING CRAMER’S RULE TO ROLLING SSCP’S

Once the 90th observation for a given id has been processed, the data could be output to a TYPE=SSCP data set for submission to PROC REG. An alternative, which turns out to be much faster, is to generate not only the rolling SSCP as above, but also the regression coefficient estimates within the same data step.

The latter task is one of solving the linear equations:

Equation 1:

\[(X'X)\hat{\beta} = (X'Y)\]

or, in more familiar terms, calculate

Equation 2:

\[\hat{\beta} = (X'X)^{-1}(X'Y).\]

This paper will not cover the matrix mathematics involved. Suffice it to say that all the elements of the matrices above can be retrieved from the SUMSQCP and SUMVARS arrays. Matrix \((X'X)\), in this example, is a 5×5 matrix taken from rows 2-5 and columns 2-5 of SUMSQCP together with a row and column using the values in SUMVARS. Matrix \((X'Y)\) is 5×1, containing values from row 1 of SUMSQCP.

Those who have had a matrix algebra class might remember confronting Cramer’s rule, which says each coefficient \(\hat{\beta}_i\) can be calculated as a ratio of determinants:

Equation 3:

\[\hat{\beta}_i = \frac{\text{det}(X'X)_i}{\text{det}(X'X)}\]

where \(\text{det}(X'X)\) is the determinant of the \((X'X)\) matrix, and \(\text{det}(X'X)_i\) is the determinant with the i'th column of \((X'X)\) replaced by the \((X'Y)\) column. This paper will not review the rules for calculating determinants, other than to note that the determinant of a k×k matrix is simply a combination of determinants of (k-1)×(k-1) sub-matrices. As an example, see the appendix for macros DET3, DET4 and DET5. DET5 calculates a determinant for a 5×5 matrix by repeated calls to DET4, which calculates determinants for 4×4 arrays by calling DET3. With these macros in hand, the data step code for generating regression coefficients is straightforward.

First generate the denominator in equation 3, i.e. the determinant of \((X'X)\):
The intercept term is generated by replacing the first column of \((X'X)\) in the numerator:

\[
\alpha = (\text{den} = \%det5(_S1, _S2, _S3, _S4, _S1, X1X1, X1X2, X1X3, X1X4, _S2, X1X2, X2X2, X2X3, X2X4, _S3, X1X3, X2X3, X3X3, X3X4, _S4, X1X4, X2X4, X3X4, X4X4)) / \text{den};
\]

The first regression coefficient is generated by replacing the second column:

\[
b1 = (\text{den} = \%det5(_SY, _S1, _S2, _S3, _S4, SYX1, X1X1, X1X2, X1X3, X1X4, SYX2, X1X2, X2X2, X2X3, X2X4, SYX3, X1X3, X2X3, X3X3, X3X4, SYX4, X1X4, X2X4, X3X4, X4X4)) / \text{den};
\]

And so on for the remaining regressors:

\[
b2 = \ldots \quad \text{(see appendix)}
\]

\[
b3 = \ldots \quad \text{(see appendix)}
\]

\[
b4 = \ldots \quad \text{(see appendix)}
\]

The complete code for applying Cramer’s Rule is in Appendix 1. Using the code above generated dramatic speed improvements: some 64.8 million windows of 60 days length for about 29,000 stocks in 14:35 – over 74,000 regressions per second.

CONCLUSIONS

There are two core tasks in generating rolling window regressions: (1) efficiently generating rolling window input for the regression, and (2) efficiently performing the regression. This paper and its predecessor (Keintz, 2012) showed that generating rolling sums-of-squares-and-cross-products is far more efficient than rolling data series, especially for larger windows. In this paper, doing that task using DATA step code is substantially faster than using PROC FCMP. In addition, using the same data step to solve for regression coefficients further improves performances.

REFERENCES:


https://www.google.com/url?q=http://www.nesug.org/Proceedings/nesug12/fi/fi08.pdf&sa=U&ei=HjboUfqtDMArA7G7xIHAcq&ved=0CBEQFjAF&client=internal-uds-cse&usg=AFQjCNMr5-dNdpWh1SvTJfvThYBMUwigiA

ACKNOWLEDGMENTS
CONTACT INFORMATION
This is a work in progress. Your comments and questions are valued and encouraged. Please contact the author at:

  Author Name:  Mark Keintz
  Company:  Wharton Research Data Services
  Address:  305 St. Leonard's Court
  Work Phone:  215.898-2160
  Fax:  215.573.6073
  Email:  mkeintz@wharton.upen.edu
APPENDIX 1: ROLLING WINDOWS REGRESSIONS WITH ARRAYS AND CRAMER’S RULE

data results (keep=id date alpha b1 b2 b3 b4);

    set orig (keep=id date Y X1 X2 X3 X4);
    by id;

    array vars {5} Y X1-X4;
    array sumvars {5} _SY _S1-_S4;

    /* vars in matrix SUMSQCP contain the rolling Sum of Squares & Cross Products */
    /* SUMSQCP is symmetric, so below diagonal cells are assigned to a placeholder*/
    array sumsqcp {5,5} _YY _YX1 _YX2 _YX3 _YX4
        XXX _X1X1 _X1X2 _X1X3 _X1X4
        XXX XXX _X2X2 _X2X3 _X2X4
        XXX XXX XXX _X3X3 _X3X4
        XXX XXX XXX XXX _X4X4 ;

    retain _SY -- _S4 _YY -- _X4X4 0;

    /* HST arrays hold (1 window) recent history of values, squares & cross-prods */
    array hstvars {90,5}  _temporary_ (90*5*0);
    array hstsqcp {90,5,5} _temporary_ (90*5*5*0);

    retain _N;  /* Counter within each ID*/

    /* Initialize arrays and counters when starting a new ID */
    if first.id then do;
        _N=0;
        do r=1 to 5;
            sumvars{r}=0;
            do c=1 to 5; sumsqcp{r,c}=0; end;
            do h=1 to 90;
                hstvars{h,r}=0;
                do c=1 to 5; hstsqcp{h,r,c}=0; end;
                end;
            end;
        end;
    end;
/* Update counters and arrays */
_N=_N+1;

/** Identify the row in HSTVARS to be replaced **/
_H=mod(_N-1,90) + 1;

do I=1 to 5;
    sumvars{I} = sumvars{I} - hstvars{_H,I} + vars{I};
    hstvars{_H,I} = vars{I};
    do J=1 to 5;
        xxx = vars{I} * vars{J};
        sumsqcp{I,J} = sumsqcp{I,J} - hstsqcp{_H,I,J} + XXX;
        hstsqcp{_H,I,J} = xxx;
    end;
end;

if _n <90 then delete; /* Wait for a complete window */

/* Now generate the regression coefficients */

denom= %det5(  90,  _S1,  _S2,  _S3,  _S4
    ,_S1,_X1X1,_X1X2,_X1X3,_X1X4
    ,_S2,_X1X2,_X2X2,_X2X3,_X2X4
    ,_S3,_X1X3,_X2X3,_X3X3,_X3X4
    ,_S4,_X1X4,_X2X4,_X3X4,_X4X4);

if denom<=0 then delete; /* Drop cases with multi-collinearity */

alpha=(%det5(_SY,  _S1,  _S2,  _S3,  _S4
    ,_YX1,_X1X1,_X1X2,_X1X3,_X1X4
    ,_YX2,_X1X2,_X2X2,_X2X3,_X2X4
    ,_YX3,_X1X3,_X2X3,_X3X3,_X3X4
    ,_YX4,_X1X4,_X2X4,_X3X4,_X4X4))/denom;

b1=(%det5( 90, _SY, _S2, _S3, _S4
    ,_S1,_YX1,_X1X2,_X1X3,_X1X4
    ,_S2,_YX2,_X2X2,_X2X3,_X2X4
    ,_S3,_YX3,_X2X3,_X3X3,_X3X4
    ,_S4,_YX4,_X3X4,_X4X4))/denom;

b2=(%det5( 90, _S1, _SY, _S3, _S4
    ,_S1,_X1X1,_YX1,_X1X3,_X1X4
    ,_S2,_X1X2,_YX2,_X2X3,_X2X4
    ,_S3,_X1X3,_YX3,_X3X3,_X3X4
    ,_S4,_X1X4,_YX4,_X3X4,_X4X4))/denom;
b3=(\%det5( 90, _S1, _S2, _SY, _S4
 ,_S1, X1X1, X1X2, YX1, X1X4
 ,_S2, X1X2, X2X2, YX2, X2X4
 ,_S3, X1X3, X2X3, YX3, X3X4
 ,_S4, X1X4, X2X4, YX4, X4X4))/denom;

b4=(\%det5( 90, _S1, _S2, _S3, _SY
 ,_S1, X1X1, X1X2, X1X3, YX1
 ,_S2, X1X2, X2X2, X2X3, YX2
 ,_S3, X1X3, X2X3, X3X3, YX3
 ,_S4, X1X4, X2X4, X3X4, YX4))/denom;

RUN;
APPENDIX 2: MACROS FOR CALCULATING DETERMINANTS FOR 3×3, 4×4 AND 5×5 MATRICES

%macro det3 (x11,x12,x13 ,x21,x22,x23 ,x31,x32,x33) / des='Determinant for a 3x3 matrix';
&x11*(&x22*&x33-&x23*&x32)
-%det3(&x21,&x33,&x34,
   &x42,&x43,&x44)
&x12*(&x21*&x33-&x23*&x31)
+&x13*(&x21*&x32-&x22*&x31)
%mend;

%macro det4(x11,x12,x13,x14 ,x21,x22,x23,x24 ,x31,x32,x33,x34 ,x41,x42,x43,x44) / des='Determinant for a 4x4 matrix';
&x11*%det3(&x22,&x23,&x24, &x32,&x33,&x34, &x42,&x43,&x44)
-%det3(&x21,&x33,&x34, &x31,&x43,&x44)
&x12*%det3(&x21,&x22,&x24, &x31,&x32,&x34, &x41,&x42,&x44)
-%det3(&x21,&x22,&x23, &x31,&x32,&x33, &x41,&x42,&x43)
%mend;

%macro det5(%macro det5(x11,x12,x13,x14,x15 ,x21,x22,x23,x24,x25 ,x31,x32,x33,x34,x35 ,x41,x42,x43,x44,x45 ,x51,x52,x53,x54,x55) / des='Determinant for a 5x5 matrix';
&x11*%det4(&x22,&x23,&x24, &x25,&x32,&x33,&x34, &x35,
   &x42,&x43,&x44,&x45,&x52,&x53,&x54,&x55))
-%det4(&x21,&x33,&x34, &x15,
   &x41,&x42,&x43,&x45,&x51,&x52,&x53,&x55))
&x12*%det4(&x21,&x23,&x24, &x25,&x31,&x33,&x34, &x35,
   &x41,&x43,&x44,&x45,&x51,&x53,&x54,&x55))
&x13*%det4(&x21,&x22,&x24, &x25,&x31,&x32,&x34, &x35,
   &x41,&x42,&x44,&x45,&x51,&x52,&x54,&x55))
-%det4(&x21,&x22,&x23, &x31,&x32,&x33,&x35,
   &x41,&x42,&x43,&x45,&x51,&x52,&x53,&x55))
+ &x15*%det4(&x21,&x22,&x23, &x24,&x31,&x32,&x33, &x35,
   &x41,&x42,&x43,&x44,&x51,&x52,&x53,&x54))
%mend;
APPENDIX 3: FCMP ROLLING SSCP SUBROUTINE

```sas
proc fcmp outlib=sasuser.temp.subr;
    deletesubr rsscp;
run;

proc fcmp outlib=sasuser.temp.subr;
    subroutine rsscp(nobs,ws,nv,_data{*,*},_rssc{*,*,*},_rsum{*,*}) varargs;
    outargs _rssc,_rsum;
    /* Arguments:
    /* NOBS:        Number of populated rows in _DATA matrix    */
    /* WS:          Window Size to develop                      */
    /* NV:          N of variables (columns) in _DATA matrix    */
    /* _DATA{*,*}   Data Items passed to this subroutine        */
    /* _RSSC{*,*,*} Rolling SSCP to return,                     */
    /*              with dimensions NOBS,NV,NV                  */
    /* _RSUM        Rolling simple sums to return (NOBS,NV)     */
    /* Generate Squares, Cross-Prods for row 1 only */
    do obs=1 to 1;
        do r=1 to nv;
            _rsum{obs,r} = _data{obs,r};
            _rssc{obs,r,r} = _data{obs,r}**2;
            if r<nv then do c=r+1 to nv;
                _rssc{obs,r,c} = _data{obs,r}*_data{obs,c};
                _rssc{obs,c,r} = _rssc{obs,r,c};
            end;
        end;
    end;
    /* Starting at obs 2 add current SQ & CP to previous total */
    do obs=2 to ws;
        do r=1 to nv;
            _rsum{obs,r} = _rsum{obs-1,r} + _data{obs,r};
            _rssc{obs,r,r} = _rssc{obs-1,r,r} + _data{obs,r}**2;
            if r<nv then do c=r+1 to nv;
                _rssc{obs,r,c} = _rssc{obs-1,r,c} + _data{obs,r}*_data{obs,c};
                _rssc{obs,c,r} = _rssc{obs,r,c};
            end;
        end;
    end;
    /* At obs ws+1 start subtracting observations leaving the window*/
    if nobs>ws then do obs=ws+1 to nobs;
        do r=1 to nv;
            _rsum{obs,r} = _rsum{obs-1,r} + _data{obs,r} - _data{obs-ws,r};
            _rssc{obs,r,r} = _rssc{obs-1,r,r} + _data{obs,r}**2 - _data{obs-ws,r}**2;
            if r<nv then do c=r+1 to nv;
    ```
_rssc{obs,r,c} = _rssc{obs-1,r,c} + _data{obs,r}*_data{obs,c} - _data{obs-ws,r}*_dataPobs-ws,c};
_rssc{obs,c,r} = _rssc{obs,r,c};
end;
end;
end;
endsub;