Solving systems of linear equations with SAS/IML

The goal of this exercise is to show you how to use SAS/IML to solve systems of linear equations. We'll do some of the initial steps contained in one of the SAS/IML examples on the web site.

1. Open the Psych 6140 web page, http://www.psych.yorku.ca/lab/psy6140 in a browser window. Navigate to SAS Examples -> Matrix Algebra with SAS/IML.
2. Select imleqn.sas
3. Start SAS/IML. There is a small library of useful IML functions you can load as follows:
```
ods listing;
proc iml;
   reset print log fuzz fw=5;
   %include iml(matlib);
```
4. The following statements define a set of 3 equations in 3 unknowns, of the form **A x** = **b**:
```
A= {1 1 -4, 1 -2 1, 1 1 1};
b= {2, 1, 0};
xx = t('X1' : 'X3');
print A '*' xx '=' b;
```
5. The SAS/IML function `r()` in the `matlib.sas` library finds the rank of a matrix. As we will see in the lecture, a system of equations is consistent (have a solution) if r(**A**) = r(**A** | **b**). Note that '`||`' is used to join matrices in IML.
```
* they are consistent, since r(A) = r(A b);
print (r(A)) (r(A || b));
```
6. For consistent equations, the solution is $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$, so the easy ways to find the solution are to use the `inv()` function or the `solve()` function:
```
x = inv(A) * b;
x = solve(A, b);
print A ' * ' x '=' (A * x) '=' b;
```
7. Another method is to use the function echelon() to reduce the matrix (**A** | **b**) to echelon form, which gives ($\mathbf{I} \mid \mathbf{A}^{-1}\mathbf{b}$)
```
*-- Echelon form of (A || b) shows solution;
r = echelon(A || b);
```
   This method also works for inconsistent equations (where `inv(A)` and `solve(A,b)` give error messages).
8. Using what you've learned so far, setup and find solution(s) to the following equations. Try `inv()`, `solve()` and `echelon()`.

$$4x_1 - 1x_2 + 3x_3 = 2$$
$$3x_1 + 5x_2 + 6x_3 = 4$$
$$1x_1 + 2x_2 + 3x_3 = 5$$

9. It may be useful to see how new functions are defined in SAS/IML. The `r()` function for matrix rank simply reduces a matrix to echelon form and counts the number of non-zero rows and columns, because rank = min (NZrows, NZcols).

```
   *-- Define a function module to find the rank of a matrix;
start r(A);
   reset noprint;
   *-- rank = number of nonzero rows/cols in echelon form;
   e = echelon(A);
   r = (e ^= 0)[,+];    *-- rows: # of non-zero elements;
   r = (r ^= 0)[+,];    *-- # non-zero rows;
   reset print;
   return (r);
finish;
```

Note the use of subscript operators `[,+]` and `[+,]` to sum a result `(e ^= 0)` across columns or rows.

10. Here is the list of functions defined in `matlib.sas`:

```
 Accessing the modules:
    %include iml(matlib);

All of these are functions, returning a value, so should be assigned
to some matrix name, e.g.,
   u = proj(y, Z);
   r = r(A);

*----- Matrix utilities -------
  row(X)    - Convert a matrix into a row vector
  col(X)    - Convert a matrix into a col vector
*----- Det, Rank, Projections -----
  r(A)      - Returns the rank of the matrix A.
  proj(y,X) - Returns the projection of vector y on matrix X.
  minor(A,i,j) - Returns the (i,j) minor of matrix A.
  cofactor(A,i,j) - Returns the (i,j) cofactor of matrix A.
*----- Statistical functions ------
  len(X)    - Returns the lengths of the columns of matrix X
  dev(x)    - Returns matrix X in deviations from its column means.
  scp(x)    - Returns sum of squares and cross-products(X`X).
  cov (x)   - Compute covariance matrix of columns of X
  corr (X)  - Compute correlations among columns of X
  median(D) - Median of each column of a matrix D
*----- Elementary row operations ----
  rowadd (X, from, to, mult) - Add 'mult'*row 'from' to row 'to'
  rowswap(X, from, to) - Interchange rows 'from' and 'to' of matrix X
  rowmult(X, row, mult) - Multiply one 'row' by a constant 'mult'
```