# Multidimensional ND Array to 2D Matrix with Wide to Long

**back to Fan's Intro Math for Econon, Matlab Examples, or MEconTools Repositories**

## 2D Matrix Wide to Long

There is a 2D matrix, the rows and columns are state variables (savings levels and shocks) for storage and graphing purposes, convert the 2D matrix where each row is a savings level and each column is a shock level to a 2D table where the first column records savings state, second column the level of shocks, and the third column stores the optimal policy or value at that particular combination of savings level and shock level.

First, generate a random 2D matrix:

```
% Create a 3D Array
it_z_n = 3;
it_a_n = 5;
% shock savings and shock array
ar_a = linspace(0.1, 50, it_a_n);
ar_z = linspace(-3, 3, it_z_n);
% function of a and z
mt_f_a_z = ar_a' + exp(ar_z);
% Display
disp(mt_f_a_z);
```

```
    0.1498    1.1000   20.1855
   12.6248   13.5750   32.6605
   25.0998   26.0500   45.1355
   37.5748   38.5250   57.6105
   50.0498   51.0000   70.0855
```

Second, from linear index to row and column index:

```
% Row and Column index for each matrix value
% Only keep non-NAN values
ar_id_isnan = isnan(mt_f_a_z);
[ar_a_idx, ar_z_idx] = ind2sub(size(mt_f_a_z), find(~ar_id_isnan));
% Display
disp([ar_a_idx, ar_a(ar_a_idx)', ar_z_idx, ar_z(ar_z_idx)']);
```

```
    1.0000    0.1000    1.0000   -3.0000
    2.0000   12.5750    1.0000   -3.0000
    3.0000   25.0500    1.0000   -3.0000
    4.0000   37.5250    1.0000   -3.0000
    5.0000   50.0000    1.0000   -3.0000
    1.0000    0.1000    2.0000         0
    2.0000   12.5750    2.0000         0
    3.0000   25.0500    2.0000         0
    4.0000   37.5250    2.0000         0
    5.0000   50.0000    2.0000         0
    1.0000    0.1000    3.0000    3.0000
    2.0000   12.5750    3.0000    3.0000
    3.0000   25.0500    3.0000    3.0000
    4.0000   37.5250    3.0000    3.0000
    5.0000   50.0000    3.0000    3.0000
```

Third, generate a 2d matrix in "table" format:

```
% Index and values
mt_policy_long = [ar_a_idx, ar_a(ar_a_idx)', ar_z_idx, ar_z(ar_z_idx)', mt_f_a_z(~ar_id_isnan)]
% Sort by a and z
mt_policy_long = sortrows(mt_policy_long, [1,3]);
```

Fourth, generate a Table with Column names:

```
% Create Table
tb_policy_long = array2table(mt_policy_long);
cl_col_names_a = {'a_idx', 'a_val', 'z_idx', 'z_val', 'pol_at_a_z'};
tb_policy_long.Properties.VariableNames = cl_col_names_a;
disp(tb_policy_long);
```

```
    a_idx    a_val     z_idx    z_val    pol_at_a_z
    _____    ______    _____    _____    __________

      1         0.1      1       -3        0.14979
      1         0.1      2        0            1.1
      1         0.1      3        3         20.186
      2      12.575      1       -3         12.625
      2      12.575      2        0         13.575
      2      12.575      3        3         32.661
      3       25.05      1       -3           25.1
      3       25.05      2        0          26.05
      3       25.05      3        3         45.136
      4      37.525      1       -3         37.575
      4      37.525      2        0         38.525
      4      37.525      3        3         57.611
      5          50      1       -3          50.05
      5          50      2        0             51
      5          50      3        3         70.086
```

## A Multidimensional ND Array with Many NaN Values

Continue with the previous exercise, but now we have more than 2 state variables.

Create a multidimensional Array with Many NaN Values. For example, we could have a dynamic lifecycle model with three endogenous varaibles, years of education accumulated, years of experiencesin blue and white collar jobs. By age 22, after starting to work at age 16, there are different possible combinations of G (schooling), X1 (white-collar), and X2 (blue-collar) jobs. These are exclusive choices in each year, so at age 16, assume that G = 0, X1 = 0 and X2 = 0. At age 16, they can choose to stay at home, school, or X1, or X2, exclusively. G, X1, X2 accumulate over time.

For each age, we can create multi-dimensional arrays with equal dimension for G, X1 and X2, to record consumption, value, etc at each element of the possible state-space. However, that matrix could have a lot of empty values.

In the example below, also has a X3 (military category).

```
% random number
rng(123);

% Max age means number of
MAX_YRS_POST16 = 3;
```

```
% store all
cl_EV = cell(MAX_YRS_POST16,1);

% Loop 1, solve BACKWARD
for it_yrs_post16=MAX_YRS_POST16:-1:1

    % Store some results, the matrix below includes all possible
    % state-space elements
    mn_ev_at_gx123 = NaN(it_yrs_post16, it_yrs_post16, it_yrs_post16, it_yrs_post16);

    % Loops 2, possibles Years attained so far as well as experiences
    for G=0:1:(it_yrs_post16-1)
        for X1=0:1:(it_yrs_post16-1-G)
            for X2=0:1:(it_yrs_post16-1-G-X1)
                for X3=0:1:(it_yrs_post16-1-G-X1-X2)

                    % Double checkAre these combinations feasible?
                    if (G+X1+X2+X3 <= it_yrs_post16)
                        % just plug in a random number
                        mn_ev_at_gx123(G+1, X1+1, X2+1, X3+1) = rand();
                    end
                end
            end
        end
    end

    % store matrixes
    cl_EV{it_yrs_post16} = mn_ev_at_gx123;

end

% Display Results
celldisp(cl_EV);
```

```
cl_EV{1} =

    0.6344



cl_EV{2} =


(:,:,1,1) =

    0.7380    0.5316
    0.5318       NaN


(:,:,2,1) =

    0.1755       NaN
       NaN       NaN


(:,:,1,2) =
```

```
    0.1825       NaN
       NaN       NaN


(:,:,2,2) =

   NaN   NaN
   NaN   NaN



cl_EV{3} =


(:,:,1,1) =

    0.6965    0.9808    0.3921
    0.3432    0.0597       NaN
    0.3980       NaN       NaN


(:,:,2,1) =

    0.5513    0.4809       NaN
    0.4386       NaN       NaN
       NaN       NaN       NaN


(:,:,3,1) =

    0.4231       NaN       NaN
       NaN       NaN       NaN
       NaN       NaN       NaN


(:,:,1,2) =

    0.2861    0.6848       NaN
    0.7290       NaN       NaN
       NaN       NaN       NaN


(:,:,2,2) =

    0.7195       NaN       NaN
       NaN       NaN       NaN
       NaN       NaN       NaN


(:,:,3,2) =

   NaN   NaN   NaN
   NaN   NaN   NaN
   NaN   NaN   NaN


(:,:,1,3) =

    0.2269       NaN       NaN
       NaN       NaN       NaN
       NaN       NaN       NaN


(:,:,2,3) =
```

```
    NaN    NaN    NaN
    NaN    NaN    NaN
    NaN    NaN    NaN


(:,:,3,3) =

    NaN    NaN    NaN
    NaN    NaN    NaN
    NaN    NaN    NaN
```

## Generate a Two Dimensional Matrix Based on ND Array for Only non-NaN Cell Values

We can generate a 2-dimensional matrix, what we can consider as a Table, with the information stored in the structures earlier. In this example, we can drop the NaN values. This matrix will be much larger in size due to explicitly storing X1, X2, X3 and G values then the ND array when most values are not NaN. But this output matrix can be much more easily interpretable and readable. When there are many many NaNs in the ND array, this matrix could be much smaller in size.

First, convert each element of the cell array above to a 2D matrix (with the same number of columns), then stack resulting matrixes together to form one big table.

```
% Create a 2D Array
for it_yrs_post16=MAX_YRS_POST16:-1:1
    % Get matrix at cell element
    mn_ev_at_gx123 = cl_EV{it_yrs_post16};
    % flaten multi-dimensional matrix
    ar_ev_at_gx123_flat = mn_ev_at_gx123(:);
    % find nan values
    ar_id_isnan = isnan(ar_ev_at_gx123_flat);
    % obtain dimension-specific index for nan positions
    [id_G, id_X1, id_X2, id_X3] = ind2sub(size(mn_ev_at_gx123), find(~ar_id_isnan));
    % generate 2-dimensional matrix (table)
    mt_ev_at_gx123 = [it_yrs_post16 + zeros(size(id_G)), ...
        (id_G-1), (id_X1-1), (id_X2-1), (id_X3-1), ...
        ar_ev_at_gx123_flat(~ar_id_isnan)];
    % stack results
    if (it_yrs_post16 == MAX_YRS_POST16)
        mt_ev_at_gx123_all = mt_ev_at_gx123;
    else
        mt_ev_at_gx123_all = [mt_ev_at_gx123_all; mt_ev_at_gx123];
    end
end
% Sort
mt_ev_at_gx123_all = sortrows(mt_ev_at_gx123_all, [1,2,3,4]);
% Create Table
tb_ev_at_gx123_all = array2table(mt_ev_at_gx123_all);
cl_col_names_a = {'YRS_POST16', 'G', 'X1', 'X2', 'X3', 'EV'};
tb_ev_at_gx123_all.Properties.VariableNames = cl_col_names_a;
disp(tb_ev_at_gx123_all);
```

```
    YRS_POST16    G    X1    X2    X3      EV
```

| | | | | | |
|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0.6344 |
| 2 | 0 | 0 | 0 | 0 | 0.738 |
| 2 | 0 | 0 | 0 | 1 | 0.18249 |
| 2 | 0 | 0 | 1 | 0 | 0.17545 |
| 2 | 0 | 1 | 0 | 0 | 0.53155 |
| 2 | 1 | 0 | 0 | 0 | 0.53183 |
| 3 | 0 | 0 | 0 | 0 | 0.69647 |
| 3 | 0 | 0 | 0 | 1 | 0.28614 |
| 3 | 0 | 0 | 0 | 2 | 0.22685 |
| 3 | 0 | 0 | 1 | 0 | 0.55131 |
| 3 | 0 | 0 | 1 | 1 | 0.71947 |
| 3 | 0 | 0 | 2 | 0 | 0.42311 |
| 3 | 0 | 1 | 0 | 0 | 0.98076 |
| 3 | 0 | 1 | 0 | 1 | 0.68483 |
| 3 | 0 | 1 | 1 | 0 | 0.48093 |
| 3 | 0 | 2 | 0 | 0 | 0.39212 |
| 3 | 1 | 0 | 0 | 0 | 0.34318 |
| 3 | 1 | 0 | 0 | 1 | 0.72905 |
| 3 | 1 | 0 | 1 | 0 | 0.43857 |
| 3 | 1 | 1 | 0 | 0 | 0.059678 |
| 3 | 2 | 0 | 0 | 0 | 0.39804 |