

Grid States, Choices and Optimal Choices Example

back to [Fan's Intro Math for Econ](#), [Matlab Examples](#), or [Dynamic Asset Repositories](#)

Generate State Grid

There many multiple individuals, each individual's value for each state space variable is different. We duplicate that by shockCount and choicecount:

```
stateCount = 2;  
shockCount = 3;  
choiceCount = 4;  
  
state1 = rand(1,stateCount)
```

```
state1 = 1x2  
    0.0571    0.6694
```

```
states1ShkDup = state1(ones(shockCount*choiceCount,1),:)
```

```
states1ShkDup = 12x2  
    0.0571    0.6694  
    0.0571    0.6694  
    0.0571    0.6694  
    0.0571    0.6694  
    0.0571    0.6694  
    0.0571    0.6694  
    0.0571    0.6694  
    0.0571    0.6694  
    0.0571    0.6694  
    0.0571    0.6694  
    0.0571    0.6694  
    0.0571    0.6694
```

```
states1ShkDup(:)
```

```
ans = 24x1  
    0.0571  
    0.0571  
    0.0571  
    0.0571  
    0.0571  
    0.0571  
    0.0571  
    0.0571  
    0.0571  
    0.0571  
    0.0571  
    0.0571  
    0.0571  
    0.0571  
    0.0571  
    0.0571  
    0.0571  
    0.0571  
    0.0571  
    0.0571  
    0.0571  
    0.0571  
    0.0571
```

Generate Choices

Generate Choice Grid, Example: Each individual has minimal protein and maximal protein they can get
Generate a evenly set grid of choices for each individual from min to max. Individual min and max choice is a function of some component of their state-space, such as wealth/income level, and choice is the quantity of good to purchase.

```
stateCount = 2;
shockCount = 3;
choiceCount = 4;
```

```
% 1. Min and Max Choices for each state
minprot_n = floor(rand(1,stateCount)*10)
```

```
minprot_n = 1×2
          7          7
```

```
maxprot_n = minprot_n + floor(rand(1,stateCount)*10)
```

```
maxprot_n = 1×2
          14          12
```

```
% 2. Choice Ratios, ratios of max-min difference
protChoiceGrid = linspace(0,1,choiceCount)
```

```
protChoiceGrid = 1×4
                0    0.3333    0.6667    1.0000
```

```
% 3. Each column is a different state.
```

```
searchMatrix = (protChoiceGrid'*(maxprot_n-minprot_n)+minprot_n(ones(choiceCount,1),:))
```

```
searchMatrix = 4×2
    7.0000    7.0000
    9.3333    8.6667
   11.6667   10.3333
   14.0000   12.0000
```

```
% 4. Each column is a different state, each set of rows is a different shock% for the state. In
```

```
searchMatrix = searchMatrix([1:choiceCount]'* ones(1,shockCount), [1:stateCount]' * ones(1,1))
```

```
searchMatrix = 12×2
    7.0000    7.0000
    9.3333    8.6667
   11.6667   10.3333
   14.0000   12.0000
    7.0000    7.0000
    9.3333    8.6667
   11.6667   10.3333
   14.0000   12.0000
    7.0000    7.0000
    9.3333    8.6667
        :
        :
```

```
searchMatrix(:)
```

```
ans = 24×1
    7.0000
    9.3333
   11.6667
   14.0000
    7.0000
    9.3333
   11.6667
   14.0000
    7.0000
    9.3333
```

:

Average Utility over Shocks

Average of Shocks, E(value) For each STATE and CHOICE, x number of shocks. Need to average over shocks; The raw value output is: STATES * SHOCKS * CHOICES; Code below turn into various things, see MATLAB CODE STRUCTURE in oneNOTE GCC working notes

```
shockCount = 2;
choiceCount = 3;
stateCount = 4;
```

```
% 1. VALUE vector (STATES * SHOCKS * CHOICES by 1), this is generated by utility% evaluation fu
valuesOri = sort(rand(choiceCount*shockCount*stateCount,1))
```

```
valuesOri = 24x1
    0.0296
    0.1141
    0.1472
    0.1514
    0.1826
    0.1936
    0.2526
    0.2911
    0.3257
    0.3352
        :
        :
```

```
% 2. CHOICES by STATES * SHOCKS (ST1 SK1, ST1 SK2; ST2 SK1, etc), each% column are values for c
values = reshape(valuesOri,[choiceCount,shockCount*stateCount])
```

```
values = 3x8
    0.0296    0.1514    0.2526    0.3352    0.5939    0.7065    0.8791    0.9204
    0.1141    0.1826    0.2911    0.3480    0.5992    0.7267    0.9001    0.9508
    0.1472    0.1936    0.3257    0.4578    0.6576    0.7792    0.9018    0.9658
```

```
% 3. SHOCKS by CHOICES * STATES (CH1 ST1, CH1 ST2; CH2 ST1, etc), each% column are two shocks f
values = reshape(values',[shockCount, choiceCount*stateCount])
```

```
values = 2x12
    0.0296    0.2526    0.5939    0.8791    0.1141    0.2911    0.5992    0.9001 ...
    0.1514    0.3352    0.7065    0.9204    0.1826    0.3480    0.7267    0.9508
```

```
% 4. AVG: 1 by CHOICES * STATES (CH1 ST1, CH1 ST2; CH2 ST1, etc), take% average over shocks for
valuesMn = mean(values,1)
```

```
valuesMn = 1x12
    0.0905    0.2939    0.6502    0.8997    0.1483    0.3196    0.6629    0.9254 ...
```

```
% 5. AVG: CHOICES * STATES. From this matrix, one can now pick maximum% utility, and match that
valuesMn = reshape(valuesMn, [stateCount, choiceCount])'
```

```
valuesMn = 3x4
    0.0905    0.2939    0.6502    0.8997
    0.1483    0.3196    0.6629    0.9254
    0.1704    0.3918    0.7184    0.9338
```

Pick Optimal Choice

```
choiceCount = 3;  
stateCount = 4;
```

```
% 1. Matrix, each column is a state, each row is a choice  
randMat = rand(choiceCount,stateCount)
```

```
randMat = 3x4  
    0.0733    0.5905    0.1731    0.1795  
    0.0550    0.8539    0.1340    0.3175  
    0.3232    0.2871    0.9947    0.5683
```

```
% 2. Maximum Value and Maximum Index  
[maxVal maxIndex] = max(randMat)
```

```
maxVal = 1x4  
    0.3232    0.8539    0.9947    0.5683  
maxIndex = 1x4  
         3         2         3         3
```

```
% 3. Linear index  
linearIdx = maxIndex + ((1:stateCount)-1)*choiceCount
```

```
linearIdx = 1x4  
         3         5         9        12
```

```
% 4. Optimal Choices  
randMat(linearIdx)
```

```
ans = 1x4  
    0.3232    0.8539    0.9947    0.5683
```