Python Codes for AEJ Paper

These are the Python codes (and outputs) as reported in Appendices A-D (and Figures 1-4) of the Accounting Educators' Journal (AEJ) paper.

In [1]:

```python
# Direct method
# Illustration based on data taken from Togo (2012)
# Case with 4 support departments (HR, PD, IT and MA) and 3 operating departments (LT, VG, ER)

import numpy as np

# Cost data and support services provided

dep_costs = np.array([800000,600000,400000,200000,2000000,1800000,1200000])
serv_perc = np.array([[-1.00,0.15,0.25,0.10,0.40,0.05,0.05],
                      [0.25,-1.00,0.20,0.15,0.35,0.00,0.05],
                      [0.20,0.10,-1.00,0.05,0.00,0.45,0.20],
                      [0.10,0.05,0.15,-1.00,0.05,0.35,0.30],
                      [0.00,0.00,0.00,0.00,1.00,0.00,0.00],
                      [0.00,0.00,0.00,0.00,0.00,1.00,0.00],
                      [0.00,0.00,0.00,0.00,0.00,0.00,1.00]])

print("Department costs:")
print(dep_costs)

# Auxiliary calculations

aux_a = serv_perc[0, 4:].sum()
aux_b = serv_perc[1, 4:].sum()
aux_c = serv_perc[2, 4:].sum()
aux_d = serv_perc[3, 4:].sum()

# Calculate the allocation percentages

serv_perc[:, 0] = 0.00
serv_perc[0, 0] = -1.00
serv_perc[:, 1] = 0.00
serv_perc[1, 1] = -1.00
serv_perc[:, 2] = 0.00
serv_perc[2, 2] = -1.00
serv_perc[:, 3] = 0.00
serv_perc[3, 3] = -1.00
serv_perc[0, 4] = (serv_perc[0, 4] / aux_a)
serv_perc[0, 5] = (serv_perc[0, 5] / aux_a)
serv_perc[0, 6] = (serv_perc[0, 6] / aux_a)
serv_perc[1, 4] = (serv_perc[1, 4] / aux_b)
serv_perc[1, 5] = (serv_perc[1, 5] / aux_b)
serv_perc[1, 6] = (serv_perc[1, 6] / aux_b)
serv_perc[2, 4] = (serv_perc[2, 4] / aux_c)
serv_perc[2, 5] = (serv_perc[2, 5] / aux_c)
serv_perc[2, 6] = (serv_perc[2, 6] / aux_c)
serv_perc[3, 4] = (serv_perc[3, 4] / aux_d)
serv_perc[3, 5] = (serv_perc[3, 5] / aux_d)
serv_perc[3, 6] = (serv_perc[3, 6] / aux_d)
print("\nAllocation percentages:")
print(serv_perc.round(2))

# Calculate the allocated costs

D = np.array([[dep_costs[0],dep_costs[0],dep_costs[0],dep_costs[0],dep_costs[0],dep_costs[0],d
ep_costs[0]],
              [dep_costs[1],dep_costs[1],dep_costs[1],dep_costs[1],dep_costs[1],dep_costs[1],d
ep_costs[1]],
              [dep_costs[2],dep_costs[2],dep_costs[2],dep_costs[2],dep_costs[2],dep_costs[2],d
ep_costs[2]],
              [dep_costs[3],dep_costs[3],dep_costs[3],dep_costs[3],dep_costs[3],dep_costs[3],d
ep_costs[3]],
              [0,0,0,0,dep_costs[4],0,0],
              [0,0,0,0,0,dep_costs[5],0],
              [0,0,0,0,0,0,dep_costs[6]]])

all_costs = np.multiply(serv_perc, D)
print("\nCost allocation (based on the direct method):")
print(all_costs.astype(int))

# Calculate the final costs of the operating departments after the allocations

col5_sum = all_costs[:, 4].sum()
col6_sum = all_costs[:, 5].sum()
col7_sum = all_costs[:, 6].sum()
print("\nFinal costs of department LT: $" + str(int(col5_sum)))
print("Final costs of department VG: $" + str(int(col6_sum)))
print("Final costs of department ER: $" + str(int(col7_sum)))
print("\nTotal costs (grand total): $" + str(int((col5_sum + col6_sum + col7_sum))))
```

```
Department costs:
[ 800000  600000  400000  200000 2000000 1800000 1200000]

Allocation percentages:
[[-1.    0.    0.    0.    0.8   0.1   0.1 ]
 [ 0.   -1.    0.    0.    0.88  0.    0.13]
 [ 0.    0.   -1.    0.    0.    0.69  0.31]
 [ 0.    0.    0.   -1.    0.07  0.5   0.43]
 [ 0.    0.    0.    0.    1.    0.    0.  ]
 [ 0.    0.    0.    0.    0.    1.    0.  ]
 [ 0.    0.    0.    0.    0.    0.    1.  ]]

Cost allocation (based on the direct method):
[[-800000       0       0       0  640000   80000   80000]
 [      0 -600000       0       0  525000       0   75000]
 [      0       0 -400000       0       0  276923  123076]
 [      0       0       0 -200000   14285  100000   85714]
 [      0       0       0       0 2000000       0       0]
 [      0       0       0       0       0 1800000       0]
 [      0       0       0       0       0       0 1200000]]

Final costs of department LT: $3179285
Final costs of department VG: $2256923
Final costs of department ER: $1563791

Total costs (grand total): $7000000
```

In [2]:

```python
# Step-down method
# Illustration based on data taken from Togo (2012)
# Case with 4 support departments (HR, PD, IT and MA) and 3 operating departments (LT, VG, ER)

import numpy as np

# Cost data and support services provided

dep_costs = np.array([800000,600000,400000,200000,2000000,1800000,1200000])
serv_perc = np.array([[-1.00,0.15,0.25,0.10,0.40,0.05,0.05],
                      [0.25,-1.00,0.20,0.15,0.35,0.00,0.05],
                      [0.20,0.10,-1.00,0.05,0.00,0.45,0.20],
                      [0.10,0.05,0.15,-1.00,0.05,0.35,0.30],
                      [0.00,0.00,0.00,0.00,1.00,0.00,0.00],
                      [0.00,0.00,0.00,0.00,0.00,1.00,0.00],
                      [0.00,0.00,0.00,0.00,0.00,0.00,1.00]])

print("Department costs:")
print(dep_costs)

# Calculate the allocation percentages

serv_perc[1:, 0] = 0.00
serv_perc[2:, 1] = 0.00
serv_perc[3:, 2] = 0.00
serv_perc[4:, 3] = 0.00

aux_a = serv_perc[1, 2:].sum()

serv_perc[1, 2] = (serv_perc[1, 2] / aux_a)
serv_perc[1, 3] = (serv_perc[1, 3] / aux_a)
serv_perc[1, 4] = (serv_perc[1, 4] / aux_a)
serv_perc[1, 5] = (serv_perc[1, 5] / aux_a)
serv_perc[1, 6] = (serv_perc[1, 6] / aux_a)

aux_b = serv_perc[2, 3:].sum()

serv_perc[2, 3] = (serv_perc[2, 3] / aux_b)
serv_perc[2, 4] = (serv_perc[2, 4] / aux_b)
serv_perc[2, 5] = (serv_perc[2, 5] / aux_b)
serv_perc[2, 6] = (serv_perc[2, 6] / aux_b)

aux_c = serv_perc[3, 4:].sum()

serv_perc[3, 4] = (serv_perc[3, 4] / aux_c)
serv_perc[3, 5] = (serv_perc[3, 5] / aux_c)
serv_perc[3, 6] = (serv_perc[3, 6] / aux_c)

dep_costs[1] = dep_costs[1] + serv_perc[0,1] * dep_costs[0]
dep_costs[2] = dep_costs[2] + serv_perc[0,2] * dep_costs[0] + serv_perc[1,2] * dep_costs[1]
dep_costs[3] = dep_costs[3] + serv_perc[0,3] * dep_costs[0] + serv_perc[1,3] * dep_costs[1] +
serv_perc[2,3] * dep_costs[2]

print("\nAllocation percentages:")
print(serv_perc.round(2))

# Calculate the allocated costs

D = np.array([[dep_costs[0],dep_costs[0],dep_costs[0],dep_costs[0],dep_costs[0],dep_costs[0],d
ep_costs[0]],
              [dep_costs[1],dep_costs[1],dep_costs[1],dep_costs[1],dep_costs[1],dep_costs[1],d
ep_costs[1]],
              [dep_costs[2],dep_costs[2],dep_costs[2],dep_costs[2],dep_costs[2],dep_costs[2],d
ep_costs[2]],
              [dep_costs[3],dep_costs[3],dep_costs[3],dep_costs[3],dep_costs[3],dep_costs[3],d
ep_costs[3]],
              [0,0,0,0,dep_costs[4],0,0],
              [0,0,0,0,0,dep_costs[5],0],
              [0,0,0,0,0,0,dep_costs[6]]])

all_costs = np.multiply(serv_perc, D)
print("\nCost allocation (based on the step-down method):")
print(all_costs.astype(int))

# Calculate the final costs of the operating departments after the allocations

col5_sum = all_costs[:, 4].sum()
col6_sum = all_costs[:, 5].sum()
col7_sum = all_costs[:, 6].sum()
print("\nFinal costs of department LT: $" + str(int(col5_sum)))
print("Final costs of department VG: $" + str(int(col6_sum)))
print("Final costs of department ER: $" + str(int(col7_sum)))
print("\nTotal costs (grand total): $" + str(int((col5_sum + col6_sum + col7_sum))))
```

```
Department costs:
[ 800000  600000  400000  200000 2000000 1800000 1200000]

Allocation percentages:
[[-1.    0.15  0.25  0.1   0.4   0.05  0.05]
 [ 0.   -1.    0.27  0.2   0.47  0.    0.07]
 [ 0.    0.   -1.    0.07  0.    0.64  0.29]
 [ 0.    0.    0.   -1.    0.07  0.5   0.43]
 [ 0.    0.    0.    0.    1.    0.    0.  ]
 [ 0.    0.    0.    0.    0.    1.    0.  ]
 [ 0.    0.    0.    0.    0.    0.    1.  ]]

Cost allocation (based on the step-down method):
[[-800000  120000  200000   80000  320000   40000   40000]
 [      0 -720000  192000  144000  335999       0   48000]
 [      0       0 -792000   56571       0  509142  226285]
 [      0       0       0 -480571   34326  240285  205959]
 [      0       0       0       0 2000000       0       0]
 [      0       0       0       0       0 1800000       0]
 [      0       0       0       0       0       0 1200000]]

Final costs of department LT: $2690326
Final costs of department VG: $2589428
Final costs of department ER: $1720244

Total costs (grand total): $6999999
```

In [3]:

```python
# Reciprocal method
# Illustration based on data taken from Togo (2012)
# Case with 4 support departments (HR, PD, IT and MA) and 3 operating departments (LT, VG, ER)

import numpy as np

# Cost data and support services provided

dep_costs = np.array([800000,600000,400000,200000,2000000,1800000,1200000])
serv_perc = np.array([[-1.00,0.15,0.25,0.10,0.40,0.05,0.05],
                      [0.25,-1.00,0.20,0.15,0.35,0.00,0.05],
                      [0.20,0.10,-1.00,0.05,0.00,0.45,0.20],
                      [0.10,0.05,0.15,-1.00,0.05,0.35,0.30],
                      [0.00,0.00,0.00,0.00,1.00,0.00,0.00],
                      [0.00,0.00,0.00,0.00,0.00,1.00,0.00],
                      [0.00,0.00,0.00,0.00,0.00,0.00,1.00]])

print("Department costs:")
print(dep_costs)

# Calculate the negative transpose matrix of the interdepartmental services

A = (serv_perc[0:4, 0:4].transpose()) * - 1.0
print("\nNegative transpose matrix:")
print(A)

# Calculate the allocated costs

B = np.array([dep_costs[0],dep_costs[1],dep_costs[2],dep_costs[3]])
C = np.linalg.solve(A, B)
print("\nReciprocated costs of the support departments:")
print(C.astype(int))

D = np.array([[C[0],C[0],C[0],C[0],C[0],C[0],C[0]],
              [C[1],C[1],C[1],C[1],C[1],C[1],C[1]],
              [C[2],C[2],C[2],C[2],C[2],C[2],C[2]],
              [C[3],C[3],C[3],C[3],C[3],C[3],C[3]],
              [0,0,0,0,dep_costs[4],0,0],
              [0,0,0,0,0,dep_costs[5],0],
              [0,0,0,0,0,0,dep_costs[6]]])

all_costs = np.multiply(serv_perc, D)
print("\nCost allocation (based on the reciprocal method):")
print(all_costs.astype(int))

# Calculate the final costs of the operating departments after the allocations

col5_sum = all_costs[:, 4].sum()
col6_sum = all_costs[:, 5].sum()
col7_sum = all_costs[:, 6].sum()
print("\nFinal costs of department LT: $" + str(int(col5_sum)))
print("Final costs of department VG: $" + str(int(col6_sum)))
print("Final costs of department ER: $" + str(int(col7_sum)))
print("\nTotal costs (grand total): $" + str(int((col5_sum + col6_sum + col7_sum))))
```

```
Department costs:
[ 800000  600000  400000  200000 2000000 1800000 1200000]

Negative transpose matrix:
[[ 1.   -0.25 -0.2  -0.1 ]
 [-0.15  1.   -0.1  -0.05]
 [-0.25 -0.2   1.   -0.15]
 [-0.1  -0.15 -0.05  1.  ]]

Reciprocated costs of the support departments:
[1275935  914985  979049  513793]

Cost allocation (based on the reciprocal method):
[[-1275935   191390   318983   127593   510374    63796    63796]
 [  228746  -914985   182997   137247   320244        0    45749]
 [  195809    97904  -979049    48952        0   440572   195809]
 [   51379    25689    77069  -513793    25689   179827   154138]
 [       0        0        0        0  2000000        0        0]
 [       0        0        0        0        0  1800000        0]
 [       0        0        0        0        0        0  1200000]]

Final costs of department LT: $2856308
Final costs of department VG: $2484197
Final costs of department ER: $1659494

Total costs (grand total): $7000000
```

In [4]:

```python
# Lattice allocation method
# Illustration based on data taken from Togo (2012)
# Case with 4 support departments (HR, PD, IT and MA) and 3 operating departments (LT, VG, ER)

import numpy as np

# Cost data, support services provided and desired accuracy

dep_costs = np.array([800000,600000,400000,200000,2000000,1800000,1200000])
serv_perc = np.array([[-1.00,0.15,0.25,0.10,0.40,0.05,0.05],
                      [0.25,-1.00,0.20,0.15,0.35,0.00,0.05],
                      [0.20,0.10,-1.00,0.05,0.00,0.45,0.20],
                      [0.10,0.05,0.15,-1.00,0.05,0.35,0.30],
                      [0.00,0.00,0.00,0.00,1.00,0.00,0.00],
                      [0.00,0.00,0.00,0.00,0.00,1.00,0.00],
                      [0.00,0.00,0.00,0.00,0.00,0.00,1.00]])
cut_off = 0.001

print("Department costs:")
print(dep_costs)

# Create the lattice allocation matrix

serv_perc[0, 0] = 0.00
serv_perc[1, 1] = 0.00
serv_perc[2, 2] = 0.00
serv_perc[3, 3] = 0.00
print("\nLattice allocation matrix:")
print(serv_perc)

# Matrix multiplication(s) needed to obtain the desired accuracy

col1_sum = serv_perc[:, 0].sum()
col2_sum = serv_perc[:, 1].sum()
col3_sum = serv_perc[:, 2].sum()
col4_sum = serv_perc[:, 3].sum()
n = 0
while (col1_sum > cut_off) | (col2_sum > cut_off) | (col3_sum > cut_off) | (col4_sum > cut_off
):
    n += 1
    serv_perc_adj = np.linalg.matrix_power(serv_perc, n)
    col1_sum = serv_perc_adj[:, 0].sum()
    col2_sum = serv_perc_adj[:, 1].sum()
    col3_sum = serv_perc_adj[:, 2].sum()
    col4_sum = serv_perc_adj[:, 3].sum()
print("\nNumber of matrix multiplications needed to obtain the desired accuracy: " + str(n))
print("\nLattice allocation matrix at the desired accuracy level:")
print(serv_perc_adj.round(2))

# Calculate the allocated costs

D = np.array([[dep_costs[0],dep_costs[0],dep_costs[0],dep_costs[0],dep_costs[0],dep_costs[0],d
ep_costs[0]],
              [dep_costs[1],dep_costs[1],dep_costs[1],dep_costs[1],dep_costs[1],dep_costs[1],d
ep_costs[1]],
              [dep_costs[2],dep_costs[2],dep_costs[2],dep_costs[2],dep_costs[2],dep_costs[2],d
ep_costs[2]],
              [dep_costs[3],dep_costs[3],dep_costs[3],dep_costs[3],dep_costs[3],dep_costs[3],d
ep_costs[3]],
              [0,0,0,0,dep_costs[4],0,0],
              [0,0,0,0,0,dep_costs[5],0],
              [0,0,0,0,0,0,dep_costs[6]]])

all_costs = np.multiply(serv_perc_adj, D)
print("\nCost allocation (based on the lattice allocation method):")
print(all_costs.astype(int))

# Calculate the final costs of the operating departments after the allocations

col5_sum = all_costs[:, 4].sum()
col6_sum = all_costs[:, 5].sum()
col7_sum = all_costs[:, 6].sum()
print("\nFinal costs of department LT: $" + str(int(col5_sum)))
print("Final costs of department VG: $" + str(int(col6_sum)))
print("Final costs of department ER: $" + str(int(col7_sum)))
print("\nTotal costs (grand total): $" + str(int((col5_sum + col6_sum + col7_sum))))
```

```
Department costs:
[ 800000  600000  400000  200000 2000000 1800000 1200000]

Lattice allocation matrix:
[[0.   0.15 0.25 0.1  0.4  0.05 0.05]
 [0.25 0.   0.2  0.15 0.35 0.   0.05]
 [0.2  0.1  0.   0.05 0.   0.45 0.2 ]
 [0.1  0.05 0.15 0.   0.05 0.35 0.3 ]
 [0.   0.   0.   0.   1.   0.   0.  ]
 [0.   0.   0.   0.   0.   1.   0.  ]
 [0.   0.   0.   0.   0.   0.   1.  ]]

Number of matrix multiplications needed to obtain the desired accuracy: 9

Lattice allocation matrix at the desired accuracy level:
[[0.   0.   0.   0.   0.54 0.27 0.19]
 [0.   0.   0.   0.   0.54 0.25 0.21]
 [0.   0.   0.   0.   0.17 0.55 0.28]
 [0.   0.   0.   0.   0.16 0.47 0.37]
 [0.   0.   0.   0.   1.   0.   0.  ]
 [0.   0.   0.   0.   0.   1.   0.  ]
 [0.   0.   0.   0.   0.   0.   1.  ]]

Cost allocation (based on the lattice allocation method):
[[    152     101     165      88  431420  218248  149822]
 [    133      87     142      76  325255  149724  124579]
 [     61      40      65      35   67913  221214  110669]
 [     24      16      26      14   31283   94524   74110]
 [      0       0       0       0 2000000       0       0]
 [      0       0       0       0       0 1800000       0]
 [      0       0       0       0       0       0 1200000]]

Final costs of department LT: $2855873
Final costs of department VG: $2483711
Final costs of department ER: $1659183

Total costs (grand total): $6998768
```