Automated Python Codes for AEJ Paper

These are automated (and therefore more advanced) versions of the Python codes (and outputs) as reported in Appendices A-D (and Figures 1-4) of the Accounting Educators' Journal (AEJ) paper. These codes can, for example, be used by a practitioner to analyze his/her real-life cost-allocation situation. Just copy-paste the codes into an Integrated Development Environment (IDE) such as Jupyter Notebook, adjust the figures and names in the part of the codes after the header "Department costs, support service percentages and department names as provided", and run the codes.

In [1]:

```
# 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

# Department costs, support service percentages and department names as 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]])
support_departments = ['HR', 'PD', 'IT', 'MA']
operating_departments = ['LT', 'VG', 'ER']

# Department costs and calculation of the number of departments

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

support = len(support_departments)
operating = len(operating_departments)
total = support + operating

# Calculation of the allocation percentages (using auxiliary calculations)

for i in range(support):
    serv_perc[:, i] = 0.00
    serv_perc[i, i] = -1.00

for i in range(support):
    aux = serv_perc[i, support:].sum()
    for j in range(support, total):
        serv_perc[i, j] = (serv_perc[i, j] / aux)

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

# Calculation of the allocated costs

D = np.array([])
tmp_list = []
for i in range(support):
    for j in range(total):
        tmp_list.append(dep_costs[i])
for i in range(support, total):
    for j in range(total):
        tmp_list.append(0)
D = np.append(D, [tmp_list])
D = D.reshape(total, total)

for i in range(support, total):
    D[i, i] = dep_costs[i]

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

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

total_costs = 0
for i in range(support, total):
    col_sum = alloc_costs[:, i].sum()
    print("\nFinal costs of department {}: $".format(operating_departments[i-support]) + str(i
nt(col_sum)), end = '')
    total_costs += col_sum

print("\n\nTotal costs (grand total): $" + str(int(total_costs)))
```

```
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]:

```
# 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

# Department cost data, support service percentages and department names as 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]])
support_departments = ['HR', 'PD', 'IT', 'MA']
operating_departments = ['LT', 'VG', 'ER']

# Department costs and calculation of the number of departments

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

support = len(support_departments)
operating = len(operating_departments)
total = support + operating

# Calculation of the allocation percentages

for x in range(1, support+1):
    y = x - 1
    serv_perc[x:, y] = 0.00

for xx in range(1, support):
    yyy = xx + 1
    zz = serv_perc[xx, yyy:].sum()
    for yy in range(yyy, total):
        serv_perc[xx, yy] = (serv_perc[xx, yy] / zz)

for i in range(1, support):
    dep_costs[i] = dep_costs[i] + serv_perc[0, i] * dep_costs[0]
    if i > 1:
        y = 1
        while support > i and y < i:
            dep_costs[i] += serv_perc[y, i] * dep_costs[y]
            y += 1

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

# Calculation of the allocated costs

D = np.array([])
tmp_list = []
for i in range(support):
    for j in range(total):
        tmp_list.append(dep_costs[i])
for i in range(support, total):
    for j in range(total):
        tmp_list.append(0)
D = np.append(D, [tmp_list])
D = D.reshape(total, total)

for i in range(support, total):
    D[i, i] = dep_costs[i]

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

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

total_costs = 0
for i in range(support, total):
    col_sum = alloc_costs[:, i].sum()
    print("\nFinal costs of department {}: $".format(operating_departments[i-support]) + str(i
nt(col_sum)), end = '')
    total_costs += col_sum

print("\n\nTotal costs (grand total): $" + str(int(total_costs)))
```

```
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]:

```
# 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

# Department cost data, support service percentages and department names as 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]])
support_departments = ['HR', 'PD', 'IT', 'MA']
operating_departments = ['LT', 'VG', 'ER']

# Department costs and calculation of the number of departments

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

support = len(support_departments)
operating = len(operating_departments)
total = support + operating

# Calculation of the negative transpose matrix of the interdepartmental services

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

# Calculation of the allocated costs

B = np.array([])
tmp_list = []
for i in range(support):
    tmp_list.append(dep_costs[i])
B = np.append(B, [tmp_list])

C = np.linalg.solve(A, B)
print("\nReciprocated costs of the support departments:")
print(C.astype(int))

D = np.array([])
tmp_list = []
for i in range(support):
    for j in range(total):
        tmp_list.append(C[i])
for i in range(support, total):
    for j in range(total):
        tmp_list.append(0)
D = np.append(D, [tmp_list])
D = D.reshape(total, total)

for i in range(support, total):
    D[i, i] = dep_costs[i]

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

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

total_costs = 0
for i in range(support, total):
    col_sum = alloc_costs[:, i].sum()
    print("\nFinal costs of department {}: $".format(operating_departments[i-support]) + str(i
nt(col_sum)), end = '')
    total_costs += col_sum

print("\n\nTotal costs (grand total): $" + str(int(total_costs)))
```

```
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]:

```
# 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

# Department cost data, support service percentages and department names as 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]])
support_departments = ['HR', 'PD', 'IT', 'MA']
operating_departments = ['LT', 'VG', 'ER']
cut_off = 0.001

# Department costs and calculation of the number of departments

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

support = len(support_departments)
operating = len(operating_departments)
total = support + operating

# Creation of the lattice allocation matrix

for i in range(support):
    serv_perc[i, i] = 0.00

print("\nLattice allocation matrix:")
print(serv_perc)

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

tmp_list = []
for i in range(support):
    tmp_list.append(serv_perc[:, i].sum())
n = 0
while (max(tmp_list) > cut_off):
    n += 1
    serv_perc_adj = np.linalg.matrix_power(serv_perc, n)
    for i in range(len(tmp_list)):
        tmp_list[i] = serv_perc_adj[:, i].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))

# Calculation of the allocated costs

D = np.array([])
tmp_list = []
for i in range(support):
    for j in range(total):
        tmp_list.append(dep_costs[i])
for i in range(support, total):
    for j in range(total):
        tmp_list.append(0)
D = np.append(D, [tmp_list])
D = D.reshape(total, total)

for i in range(support, total):
    D[i, i] = dep_costs[i]

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

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

total_costs = 0
for i in range(support, total):
    col_sum = alloc_costs[:, i].sum()
    print("\nFinal costs of department {}: $".format(operating_departments[i-support]) + str(i
nt(col_sum)), end = '')
    total_costs += col_sum

print("\n\nTotal costs (grand total): $" + str(int(total_costs)))
```

```
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
```