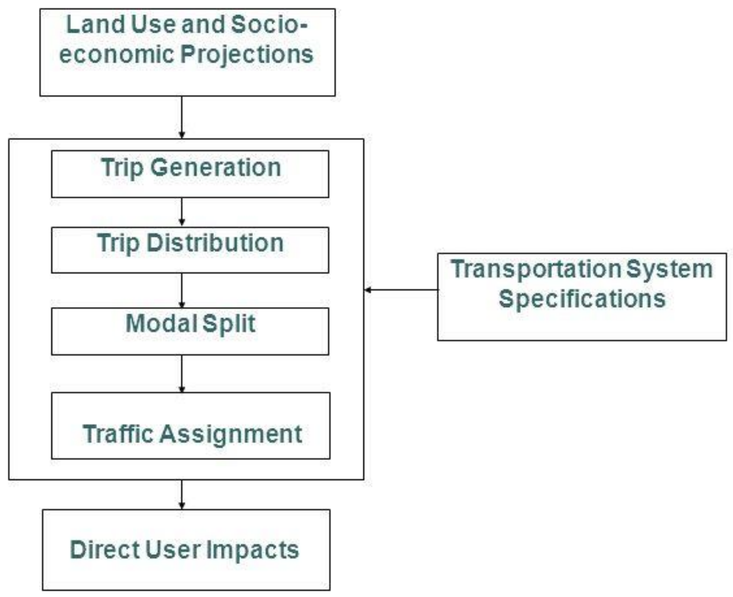
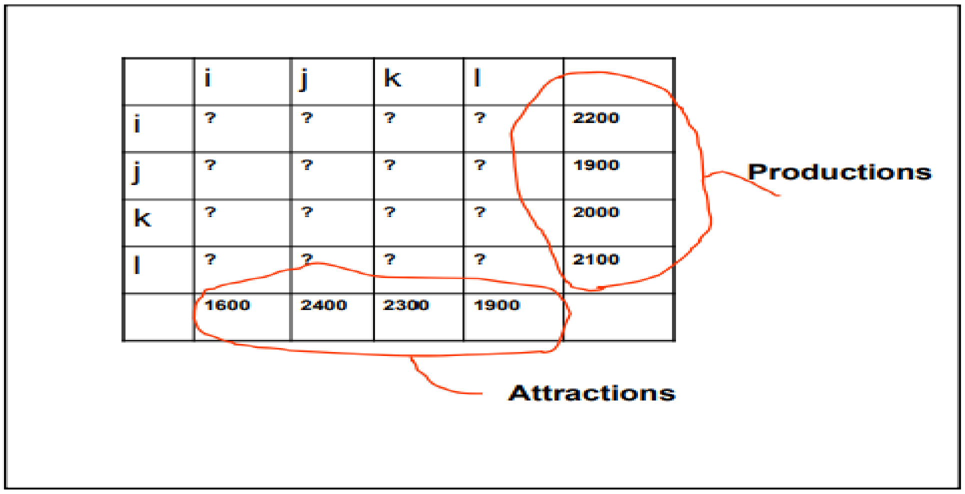
This presentation to learn how to turn Furness method that used in trip distribution using python

Overall Procedure

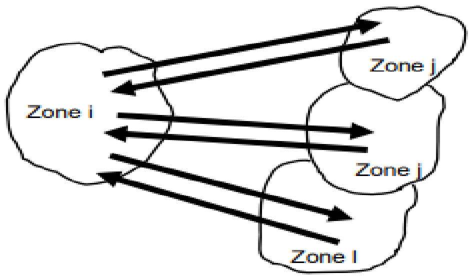




	i	j	k	l	
i	?	?	?	?	2200
j	?	?	?	?	1900
k	?	?	?	?	2000
l	?	?	?	?	2100
	1600	2400	2300	1900	

Productions

Attractions



Furness' Algorithm (1965)

$$t_{ij}^{f} = gf_{ij} * t_{ij}^{c}$$

$$t_{ij}^{f} = a_i * b_j * t_{ij}^{c}$$

a_i = Growth factor due to productions
b_j = Growth factor due to attractions

Step 1
Assume b's =1 and solve for a's that satisfies the production constraints
Step 2
With the latest a's, solve for b's that satisfies the attraction constraints.
Step 3
keeping the b's fixed, solve for a's and repeat steps (2) and (3) until convergence.

-First method used to get the best values of a and b that achieve the best accuracy with the current cells (zones)

Code :

```
import numpy as np

originalOD = np.array([[10,60,80,50],
[80,20,100,50],[20,130,10,50],[100,80,60,20]])

print("the originalOD is :\n",originalOD)

# sum A and B

targetD=np.array([420,435,250,515])
targetO=np.array([300,250,420,650])

B = np.array([1,1,1,1])

Number_Of_Iterations = 0
Convergence=0
while  True :

    newOD=np.multiply(originalOD,B)
    newO=np.sum(newOD,axis=1)
    A = targetO / newO

    newOD=np.multiply(originalOD,A)
    newD = np.sum(newOD, axis=0)
    B = targetD / newD

    print("Number Of Iterations ",Number_Of_Iterations)

    x1,x2,x3,x4= np.split(A,[1,2,3])
    y1,y2,y3,y4= np.split(B,[1,2,3])

    print(f"a1 = {x1} \na2 = {x2} \na3 = {x3} \na4 = {x4} \n")
    print(f"b1 = {y1} \nb2 = {y2} \nb3 = {y3} \nb4 = {y4} \n")
    # print(f"a2 = {x2}")
    # print(f"b1 = {y1}")
    # print(f"b2 = {y2}")
    Number_Of_Iterations +=1
    Convergence = (sum(newO)/sum(targetO)+sum(newD)/sum(targetD))/2

    print('Converged:', Convergence)
    if Convergence<1.04 and Convergence>0.999999:
        break
    else:
        continue
```

-the second method used to predict the zones in the future with best values of a and b that means that the current will be changed in this situation (updated values)

Code:

```
# import numpy to deal with matrices
import numpy as np
# declare the original matrix in this case
originalOD = np.array([[200,700], [300,100]])

print("the originalOD is :\n",originalOD)

# declare the future of original and the future of destination

targetD=np.array([1100,1500])
targetO=np.array([1800,900])
# assume that B's = 1 to clac first value of A

Convergence=0
new_Od=originalOD
newD=np.sum(new_Od,axis=0)
newO=np.sum(new_Od,axis=1)
Number_Of_Iterations =0
while  True :
    # do next
    #A = future A / sum of mlutiply by B
    B = targetD / newD
    A = targetO / newO
    new_Od = np.multiply(new_Od, A)
    newD = np.sum(new_Od, axis=0)

    new_Od=np.multiply(new_Od,B)
    newO=np.sum(new_Od,axis=1)

    # do next
    # B = future B / sum of mlutiply by A
    print("Number Of Iterations ",Number_Of_Iterations)
    # split the array that contain the all value of A to a1,a2,...........,n
    x1,x2= np.split(A,[1])
    # split the array that contain the all value of B to a1,a2,...........,n
    y1,y2= np.split(B,[1])

    print(f"a1 = {x1} \na2 = {x2} \n")
    print(f"b1 = {y1} \nb2 = {y2} \n")
    Number_Of_Iterations +=1

    Convergence = (sum(newO)/sum(targetO)+sum(newD)/sum(targetD))/2
    print('Converged:', round(Convergence,3))
    if Convergence < 1.01 and Convergence > 0.99:
        break
    else:
        continue
```

Thanks