We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

5,000
Open access books available

125,000
International authors and editors

140M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

Chapter

A Universal Methodology for Generating Elevator Passenger Origin-Destination Pairs for Calculation and Simulation

Lutfi Al-Sharif

Abstract

The origin-destination matrix is a two-dimensional matrix that describes the probability of a passenger travelling from one floor in the building to another. It is a two-dimensional square matrix. The row index denotes the origin floor and the row index denotes the destination floor for the passenger journey. A previous chapter described the methodology for constructing the origin-destination matrix (OD matrix) from the user requirements. However, that chapter placed the restriction that any floor must either be assigned as an entrance floor or an occupant floor, but not both. This chapter relaxes this restriction and shows a method for developing the origin-destination matrix that allows any floor to either be an entrance floor; an occupant floor; or both. The origin destination matrix can be compiled using three sets of parameters: the mix of traffic (incoming traffic, outgoing traffic, inter-floor traffic; and inter-entrance traffic); the floor populations; and the entrance percentage bias (i.e., the relative strength of the arrivals at the entrance floors). The origin-destination matrix can be used for the generation of random passenger origin-destination pairs (which is necessary when using the Monte Carlo Simulation (MCS) method to calculate the round-trip time or in elevator traffic software).

Keywords: elevator, lift, round trip time, incoming traffic, outgoing traffic, inter-floor traffic, inter-entrance traffic, general traffic conditions, Monte Carlo simulation, origin-destination matrix

1. Introduction

In an elevator traffic system within a building, the origin-destination matrix is a compact-concise tool that is used to clearly describe the probability of a passenger travelling from one floor in the building to another. It is a two-dimensional square matrix. The diagonal elements of the matrix are equal to zero, since rational passenger behaviour is assumed (i.e. no passengers will travel from a floor to the same floor). Moreover, the sum of all the elements within the matrix is equal to one (representing the universal set in probability). The row index of the matrix denotes the origin floor of the passenger's journey, whilst the column row denotes the destination floor of the passenger's journey.

It is worth noting that all the events in the *OD* matrix are mutually exclusive (i.e., if one of them takes place in a round trip, the others cannot take place in the same round trip). As an example, if a passenger chooses to go from the third floor to the seventh floor in a round trip, he/she cannot also go from the eighth floor to the second floor in the same round trip.

The general format for an *OD* matrix is shown below. All the diagonal elements are equal to zero, as it is assumed that passengers are rational and would not travel from a floor to the same floor.

$$OD_{adj} = \begin{bmatrix} 0 & p_{1\cdot 2} & .. & .. & p_{1\cdot N-1} & p_{1\cdot N} \\ p_{2\cdot 1} & 0 & .. & .. & p_{2\cdot N-1} & p_{2\cdot N} \\ .. & .. & .. & .. & .. & .. \\ .. & .. & .. & .. & .. & .. \\ p_{N-1\cdot 1} & p_{N-1\cdot 2} & .. & .. & 0 & p_{N-1\cdot N} \\ p_{N\cdot 1} & p_{N\cdot 2} & .. & .. & p_{N\cdot N-1} & 0 \end{bmatrix}$$

The origin-destination matrix is critical for the elevator traffic design process, specifically for the following two functions:

1. The origin-destination matrix is used to generate passengers with origin and destination pairs to be used in finding the round-trip time using Monte Carlo simulation. This can be done for both calculation and simulation [1].

2. The origin–destination matrix can also be used to derive the probabilities of any type of event taking place in a round trip (e.g., the probability of a journey taking place between the third and sixth floors without stopping at the fourth and fifth floors). Deriving these probabilities is critical to deriving equations for evaluating the round-trip time.

The final origin-destination matrix must obey a number of rules, listed below:

1. The sum of all the elements in the *OD* matrix should add up to 1.

2. All the elements of the diagonal of the matrix should be equal to 0.

It is worth noting that work has been carried out in trying to estimate the origin-destination traffic from elevator movements in the building ([2–5]). The outcome can be used in a number of ways, including generating virtual passenger traffic [6] or deciding on the suitable group control algorithm to adopt during that period of time [7].

Previous work has shown how the origin destination matrix can be derived ([8, 9]). However, those two pieces of work have assumed that any floor must either be an entrance floor or an occupant floor, but not both. The work presented in this paper relaxes this requirement and allows any floor to be both an entrance floor and an occupant floor.

Examples of traffic mix conditions that are believed to be representative of the lunch-time peak traffic conditions in many modern office buildings include: 40%:40%:20% [10]; 45%:45%:10% [11]; 42%, 42%, 16% [12]; incoming, outgoing and inter-floor traffic, respectively.

In addition, it is possible to calculate the round-trip time for general traffic conditions under the Poisson arrival process assumption ([13, 14]) or the plentiful-passenger-supply assumption ([15, 16]).

This framework can then be combined with other design methodologies in order to be used as a comprehensive universal elevator traffic system design tool, whereby the average passenger waiting time and the average passenger travelling times can also be added as user requirements, and outputs such as the car capacity, the elevator speeds can be provided as outputs of the design.

Existing software packages [17] employ a slightly different method for generating passenger origin-destination pairs, which leads to marginally different results.

Section 2 describes the two types of floors. Section 3 describes the possible modes of traffic in a building. Section 4 describes how the traffic can be described in a building. Section 5 shows how the universal origin destination matrix can be derived. A numerical example is given in Section 6. Section 7 discusses the issue of verification to ensure the correctness of the final OD matrix. Conclusions are drawn in Section 8.

2. Types of floors

This section looks at the classification of floors. Any floor in a building can either be described as an entrance floor or an occupant floor (or both as presented in this paper). An entrance floor (referred to in short as *ent* floor) is a floor through which passengers can either enter or exit the building. An entrance floor can also be referred to an entrance/exit floor, as it becomes an exit floor when the traffic is out-going.

An occupant floor is a floor through which passengers cannot enter or exit the building, but where they will reside during their stay in the building (referred to in short as an *occ* floor). Based on the assumption above, all floors in the building would be denoted as either entrance floors or occupant floors (or both as is allowed in this paper). In some buildings, a floor could simultaneously be an entrance floor and an occupant floor (e.g., ground floor that has a population on it).

An example of a building that is represented in this form is shown below in **Table 1**. Each floor is either an entrance/exit floor or an occupant floor, although one floor is designated with both functions (entrance and occupant) which is the ground floor. The percentage of passengers entering the building via an entrance floor is denoted as the percentage arrival rate [$Pr_{arr}(i)$]. The population of a floor expressed as a percentage of the total building population is denoted as the percentage population [$(U(i)/U)$].

Where a floor is an entrance floor, it has a nonzero value for its percentage arrival rate and a zero-percentage population. Where a floor is an occupant floor, it has a nonzero percentage population but a zero-percentage arrival rate. Where it has mixed designation, it can have both.

Floor notation	Floor name	Percentage arrival rate	Population	Population percentage
N	*L5*	0	50	0.10
N-1	*L4*	0	50	0.10
...	*L3*	0	75	0.15
...	*L2*	0	75	0.15
...	*L1*	0	100	0.20
3	*G*	0.7	150	0.30
2	*B1*	0.2	0	0.00
1	*B2*	0.1	0	0.00

Table 1.
Representation of traffic in a building.

It is customary for the entrance floors to be contiguous and for the occupant floors to be contiguous, but this is not necessary. An example of a noncontiguous floor is where a restaurant is located on the top floor of a building and is classified as an entrance/exit floor (as it is functionally external to the building albeit not physically external). The design methodology presented later in this paper can cope with the general case where the entrance floors are noncontiguous and the occupant floors are noncontiguous.

Table 1 shows a building with three entrances with unequal percentage arrival rates (0.7 ground floor denoted as G; 0.2 from the basement denoted as B1; 0.1 from the basement denoted as B2). There are six occupant floors (G and L1 to L5). They have unequal population percentages.

The lowest floor in the building is denoted as floor 1 and the topmost floor as floor *N*. The following convention is followed in describing the values of arrival percentages and populations for the floors:

$Pr_{arr}(i)$ is used to denote the arrival percentage of the i^{th} floor, where i runs from 1 to N.

$U(i)/U$ is used to denote the percentage population of the i^{th} floor, where i runs from 1 to N, where $U(i)$ is the population of the i^{th} floor and U is the total building population.

The representation of traffic in a building as shown in **Table 1** is the default format and represents pure incoming traffic into the building. Under such a traffic condition, all passengers would be entering the building form the entrance floors and heading to the occupant floors.

A general format for representing the traffic in a building is shown in **Table 2** below. As a generalisation, the term arrival can be extended to arrival/departure to cover both passengers entering the building under incoming traffic conditions and passengers leaving the building under out-going traffic conditions. By setting a population percentage for a floor to zero, it is an indication that it is an entrance floor; by setting an arrival percentage for a floor to zero, it is an indication that it is an occupant floor. By setting a value to the population percentage and a value to the percentage arrival rate, it is an indication that it has dual function.

It is worth noting that the summation of the percentage arrival/departure rates is 1 and the summation of all the building percentage populations is 1 as shown in Eqs. (1) and (2) below.

$$\sum_{i=1}^{N} \mathrm{Pr}_{arr}(i) = 1 \tag{1}$$

Floor	Percentage arrival/Departure	Percentage population
N	$Pr_{arr}(N)$	$(U(N)/U)$
N-1	$Pr_{arr}(N-1)$	$(U(N-1)/U)$
...	...	...
...	...	...
...	...	...
2	$Pr_{arr}(2)$	$(U(2)/U)$
1	$Pr_{arr}(1)$	$(U(1)/U)$

Table 2.
General representation format for a building.

Start floor (origin)	End floor (destination)	Type of traffic	Description
Entrance/exit	Occupant	Incoming traffic	Passengers arriving at the building
Occupant	Entrance/exit	Outgoing traffic	Passengers leaving the building
Occupant	Occupant	Inter-floor traffic	Passengers moving within the building (restaurants, meeting rooms)
Entrance/exit	Entrance/exit	Inter-entrance traffic	Usually very small

Table 3.
Types of traffic.

$$\sum_{i=1}^{N}\left(\frac{U(i)}{U}\right)=1 \tag{2}$$

Two row vectors (arrays) can be developed based on the arrival percentages and the floor population percentages, as shown below. The first row vector is the percentage arrivals, denoted as Pr_{arr}:

$$\boldsymbol{Pr_{arr}} = [Pr_{arr}(1)\ \ Pr_{arr}(2)\ \ \ldots\ \ \ldots\ \ \ldots\ \ Pr_{arr}(N-1)\ \ Pr_{arr}(N)] \tag{3}$$

The convention that will be used in this paper is that the indexing will start from the lowest floor in the building and increase upwards. The percentage population can be also be organised into a row vector:

$$\boldsymbol{U_{nor}} = [\,U(1)/U \quad U(2)/U \quad \ldots \quad \ldots \quad \ldots \quad U(N-1)/U \quad U(N)/U\,] \tag{4}$$

The "nor" subscript stands for "normalised".

3. Types of traffic

This section classifies the possible types of journeys and thus the possible types of traffic. Every passenger journey must logically have an origin and a destination. Considering that any floor can either be an entrance/exit floor or an occupant floor, there can exist in theory four types of journeys depending on the classification of the origin and destination floors for each journey.

A journey that starts from an entrance/exit floor and terminates at an occupant floor is denoted as an incoming traffic journey. A journey that starts from an occupant floor and terminates at an entrance/exit floor is denoted as an outgoing traffic journey. A journey that starts from an occupant floor and terminates at an occupant floor is denoted as an inter-floor journey. A journey that starts from an entrance/exit floor and terminates at an entrance/exit floor is denoted as an inter-entrance journey. These four types of traffic are listed in **Table 3** below.

4. Description of the traffic in a building

It has become customary to describe the prevailing traffic in a building at any one point in time as a mixture of the four types of traffic described in the previous section.

The percentage of the traffic that is incoming at any one point in time is denoted as *ic*; the percentage of the traffic that is outgoing at any one point in time is denoted as *og*; and the traffic that is inter-floor at any one point in time is denoted as *if*; and the traffic that is inter-entrance is denoted as *ie*. The combination of these four numbers can be used to describe the traffic mix as shown below (where any of these parameters can vary between 0 and 1):

$$ic : og : if : ie$$

As expected, the sum of all four numbers should add up to 1 as shown in Eq. (5) below. Thus, assigning values for three of these numbers automatically sets the value of the fourth parameter.

$$ic + og + if + ie = 1 \tag{5}$$

As an example, one suggested composition of the lunchtime traffic conditions can be described by the following representation [11]:

$$ic : og : if : ie \text{ as } 0.45 : 0.45 : 0.10 : 0.0 \text{ respectively}.$$

5. The origin-destination (*OD*) matrix

At the end of Section 2, the percentage floor populations were compiled in a concise normalised format in the shape of a row vector denoted as $\boldsymbol{U_{nor}}$. In addition, the percentage arrivals were compiled in a concise format in the shape of a row vector denoted as ***Prarr***.

In order to compile the overall final origin destination, it is first necessary to develop the four origin-destination matrices that contain the contribution from the four types of traffic (incoming, outgoing, inter-floor and inter-entrance). Once these four matrices have been fully developed, the final overall origin-destination matrix is found by adding these four matrices.

5.1 Finding the initial values of the four OD matrices

The first matrix (the incoming traffic matrix) is obtained by multiplying the transpose of the percentage arrival vector ($\boldsymbol{Pr_{arr}}$) by the normalised population vector ($\boldsymbol{U_{nor}}$). As the dimensions of the percentage arrival row vector is 1 by N, when it is transposed, it dimensions become N by 1. The dimensions of the population normalised row vector is 1 by N. When an N by 1 vector is multiplied by a 1 by N vector, this results in an N by N matrix.

$$OD_{ic\_ini} = Pr_{arr}{}^{T} \bullet U_{nor} \tag{6}$$

The second matrix (the outgoing traffic matrix) is obtained by multiplying the transpose of the normalised population vector ($\boldsymbol{U_{nor}}$) by the percentage arrival vector ($\boldsymbol{Pr_{arr}}$) as shown in Eq. (7) below.

$$OD_{og\_ini} = U_{nor}{}^{T} \bullet Pr_{arr} \tag{7}$$

The third matrix (the inter-floor traffic matrix) is obtained by multiplying the transpose of the normalised population vector ($\boldsymbol{U_{nor}}$) by the normalised population vector ($\boldsymbol{U_{nor}}$) as shown in Eq. (8) below.

$$OD_{if\_ini} = U_{nor}{}^{T} \bullet U_{nor} \tag{8}$$

The fourth matrix (the inter-entrance traffic matrix) is obtained by multiplying the transpose of the percentage arrival vector ($\boldsymbol{Pr_{arr}}$) by the percentage arrival vector ($\boldsymbol{Pr_{arr}}$) as shown in Eq. (7) below.

$$OD_{ie\_ini} = Pr_{arr}{}^{T} \bullet Pr_{arr} \tag{9}$$

where the subscript T denotes the transpose of a matrix.

It is worth noting that each of these matrices that result from the multiplication have a total sum of all the elements that is equal to 1.

5.2 Adjusting the four OD matrices

Once the four initial Origin-Destination matrices have been produced by multiplying the relevant vectors, the next step is to remove the irrational traffic (travelling from a floor back to the same floor). This is done in three steps as discussed below:

a. Finding the value of the adjusting factor, M, for each matrix. The value of M is necessary in order to re-adjust the sum of the matrix back to 1, later on. M is calculated as shown below:

$$M = 1 - \left(\sum_{i=1}^{N} (pr_{ii}) \right) \tag{10}$$

b. The second step is to zero all the diagonal elements of the four matrices. This removes the irrational behaviour of a passenger going from a floor back to the same floor.

$$p_{ii} = 0 \quad for \quad i = j \tag{11}$$

c. But as the diagonal items have been zeroed, the sum of all the elements in the matrix no longer adds up to 1. Thus, the third step is to re-adjust the remaining nonzero elements of the matrices in order to restore the sum of all elements in the matrix back to 1. This is done by dividing each element in the matrix by the value of M, found in Eq. (10) above.

Once these three steps have been carried out, the resulting four OD matrices become the final matrices.

$$OD_{ic\_fin}, OD_{og\_fin}, OD_{if\_fin}, OD_{ie\_fin}$$

5.3 Finding the final OD matrix from the four OD matrices

The final origin–destination matrix can now be calculated by adding the weighted sum of the four final matrices. Each one is multiplied by the percentage of the traffic that is represents. This is shown in Eq. (12) below.

$$OD_{fin} = ic \bullet OD_{ic\_fin} + og \bullet OD_{og\_fin} + if \bullet OD_{if\_fin} + ie \bullet OD_{ie_{fin}} \tag{12}$$

This matrix can also be referred to as the "*normalised* origin-destination matrix". It is referred to as *normalised* as it only depends on the arrival percentages of the floors, the population percentages of the floors and the mix of traffic. It does not depend on the passenger arrival intensity. The matrix in this form can now be used in order to generate random passenger origin–destination pairs for evaluating the round-trip time using the Monte Carlo Simulation (*MCS*) method. It can also be used to evaluate the round-trip time using formula by calculation. The sum of all the elements of the matrix is equal to 1. The diagonal element of the matrix must be equal to zero.

6. Numerical example on evaluating the universal origin-destination matrix and generating passenger origin-destination pairs from it

This section presents a practical numerical example on developing the universal original-destination matrix.

6.1 The arrangement

A building has three entrance floors (B2, B1 and G) and six occupant floors (G, L1, L2, L3, L4 and L5). The ground floor (G) has a dual function, hence that is why is appears in both lists. The percentage arrival rates from the three entrances are 0.1, 0.2 and 0.7 from B2, B1 and G respectively. The percentage populations of the occupant floors are: 30%, 20%, 15%, 15%, 10% and 10% for the occupant floors G, L1, L2, L3, L4 and L5, respectively. All the building details are shown in **Table 4**. It is required that the overall origin–destination matrix be developed based on a traffic mix of 40%:30%:20%:10% of incoming, outgoing, inter-floor and inter-entrance traffic, respectively.

It is worth noting that the ground floor in this case is a dual function floor: it is an entrance floor and an occupant floor simultaneously. This is the sort of example that requires a universal OD matrix, whereby any floor can be an entrance and an occupant floor at the same time.

The next step is to create the initial values of the four distinct matrices: the incoming traffic matrix, the outgoing traffic matrix, the inter-floor matrix and the inter-entrance. These are shown below:

#	Percentage arrival	Population	Percentage population
L5	0	50	0.10
L4	0	50	0.10
L3	0	75	0.15
L2	0	75	0.15
L1	0	100	0.2
G	0.7	150	0.3
B1	0.2	0	0
B2	0.1	0	0

Table 4.
Generalised representation of traffic for a building depending on the traffic mix.

6.2 The incoming traffic matrix

The first matrix to be developed is the incoming traffic matrix, as shown below.

OD_{ic_ini}

		0	0	0.3	0.2	0.15	0.15	0.1	0.1
		B2	**B1**	**G**	**L1**	**L2**	**L3**	**L4**	**L5**
0.1	**B2**	0	0	0.03	0.02	0.015	0.015	0.01	0.01
0.2	**B1**	0	0	0.06	0.04	0.03	0.03	0.02	0.02
0.7	**G**	0	0	0.21	0.14	0.105	0.105	0.07	0.07
0	**L1**	0	0	0	0	0	0	0	0
0	**L2**	0	0	0	0	0	0	0	0
0	**L3**	0	0	0	0	0	0	0	0
0	**L4**	0	0	0	0	0	0	0	0
0	**L5**	0	0	0	0	0	0	0	0

The value of M for the incoming traffic matrix is calculated as shown below:

$$M = 1 - \left(\sum_{i=1}^{N} (pr_{ii}) \right) = 0.79 \tag{13}$$

Zeroing the diagonal and then dividing all the elements by M gives the final incoming traffic matrix, shown below.

OD_{ic_fin}

0	**B2**	**B1**	**G**	**L1**	**L2**	**L3**	**L4**	**L5**
B2	0	0	0.01519	0.010127	0.007595	0.007595	0.005063	0.005063
B1	0	0	0.03038	0.020253	0.01519	0.01519	0.010127	0.010127
G	0	0	0	0.070886	0.053165	0.053165	0.035443	0.035443
L1	0	0	0	0	0	0	0	0
L2	0	0	0	0	0	0	0	0
L3	0	0	0	0	0	0	0	0
L4	0	0	0	0	0	0	0	0
L5	0	0	0	0	0	0	0	0

6.3 The outgoing traffic matrix

The same process is applied in order to calculate outgoing traffic matrix.

OD_{og_ini}

		0	0	0.3	0.2	0.15	0.15	0.1	0.1
		B2	**B1**	**G**	**L1**	**L2**	**L3**	**L4**	**L5**
0.1	**B2**	0	0	0	0	0	0	0	0

0.2	**B1**	0	0	0	0	0	0	0	0
0.7	**G**	0.03	0.06	0.21	0	0	0	0	0
0	**L1**	0.02	0.04	0.14	0	0	0	0	0
0	**L2**	0.015	0.03	0.105	0	0	0	0	0
0	**L3**	0.015	0.03	0.105	0	0	0	0	0
0	**L4**	0.01	0.02	0.07	0	0	0	0	0
0	**L5**	0.01	0.02	0.07	0	0	0	0	0

The value of M for the incoming traffic matrix is calculated as shown below:

$$M = 1 - \left(\sum_{i=1}^{N} (pr_{ii}) \right) = 0.79 \tag{14}$$

Zeroing the diagonal and then dividing all the elements by M gives the final incoming traffic matrix, shown below.

OD_{og_fin}

0	**B2**	**B1**	**G**	**L1**	**L2**	**L3**	**L4**	**L5**
B2	0	0	0	0	0	0	0	0
B1	0	0	0	0	0	0	0	0
G	0.011392405	0.02278481	0	0	0	0	0	0
L1	0.007594937	0.015189873	0.053164557	0	0	0	0	0
L2	0.005696203	0.011392405	0.039873418	0	0	0	0	0
L3	0.005696203	0.011392405	0.039873418	0	0	0	0	0
L4	0.003797468	0.007594937	0.026582278	0	0	0	0	0
L5	0.003797468	0.007594937	0.026582278	0	0	0	0	0

6.4 The inter-floor traffic matrix

The same process is applied in order to calculate inter-floor traffic matrix.

OD_{if_ini}

		0	**0**	**0.3**	**0.2**	**0.15**	**0.15**	**0.1**	**0.1**
		B2	**B1**	**G**	**L1**	**L2**	**L3**	**L4**	**L5**
0.1	**B2**	0	0	0	0	0	0	0	0
0.2	**B1**	0	0	0	0	0	0	0	0
0.7	**G**	0	0	0.09	0.06	0.045	0.045	0.03	0.03
0	**L1**	0	0	0.06	0.04	0.03	0.03	0.02	0.02
0	**L2**	0	0	0.045	0.03	0.0225	0.0225	0.015	0.015
0	**L3**	0	0	0.045	0.03	0.0225	0.0225	0.015	0.015
0	**L4**	0	0	0.03	0.02	0.015	0.015	0.01	0.01
0	**L5**	0	0	0.03	0.02	0.015	0.015	0.01	0.01

The value of M for the incoming traffic matrix is calculated as shown below:

$$M = 1 - \left(\sum_{i=1}^{N} (pr_{ii}) \right) = 0.805 \tag{15}$$

Zeroing the diagonal and then dividing all the elements by M gives the final incoming traffic matrix, shown below.

OD_{if_fin}

0	**B2**	**B1**	**G**	**L1**	**L2**	**L3**	**L4**	**L5**
B2	0	0	0	0	0	0	0	0
B1	0	0	0	0	0	0	0	0
G	0	0	0	0.014907	0.01118	0.01118	0.007453	0.007453
L1	0	0	0.014907	0	0.007453	0.007453	0.004969	0.004969
L2	0	0	0.01118	0.007453	0	0.00559	0.003727	0.003727
L3	0	0	0.01118	0.007453	0.00559	0	0.003727	0.003727
L4	0	0	0.007453	0.004969	0.003727	0.003727	0	0.002484
L5	0	0	0.007453	0.004969	0.003727	0.003727	0.002484	0

6.5 The inter-entrance traffic matrix

The same process is applied in order to calculate inter-entrance traffic matrix.

OD_{if_ini}

		0	**0**	**0.3**	**0.2**	**0.15**	**0.15**	**0.1**	**0.1**
		B2	**B1**	**G**	**L1**	**L2**	**L3**	**L4**	**L5**
0.1	**B2**	0.01	0.02	0.07	0	0	0	0	0
0.2	**B1**	0.02	0.04	0.14	0	0	0	0	0
0.7	**G**	0.07	0.14	0.49	0	0	0	0	0
0	**L1**	0	0	0	0	0	0	0	0
0	**L2**	0	0	0	0	0	0	0	0
0	**L3**	0	0	0	0	0	0	0	0
0	**L4**	0	0	0	0	0	0	0	0
0	**L5**	0	0	0	0	0	0	0	0

The value of M for the incoming traffic matrix is calculated as shown below:

$$M = 1 - \left(\sum_{i=1}^{N} (pr_{ii}) \right) = 0.46 \tag{16}$$

Zeroing the diagonal and then dividing all the elements by M gives the final incoming traffic matrix, shown below.

OD_{if_fin}

0	B2	B1	G	L1	L2	L3	L4	L5
B2	0	0.00434783	0.015217	0	0	0	0	0
B1	0.004348	0	0.030435	0	0	0	0	0
G	0.015217	0.03043478	0	0	0	0	0	0
L1	0	0	0	0	0	0	0	0
L2	0	0	0	0	0	0	0	0
L3	0	0	0	0	0	0	0	0
L4	0	0	0	0	0	0	0	0
L5	0	0	0	0	0	0	0	0

6.6 Finding the final overall OD matrix

The final step is to combine all the four matrices by multiplying each matrix by the associated traffic percentage mix and adding them up. This gives the final OD matrix shown below.

OD_{fin}

	B2	B1	G	L1	L2	L3	L4	L5
B2	0	0.004348	0.030407	0.010127	0.007595	0.007595	0.005063	0.005063
B1	0.004348	0	0.060815	0.020253	0.01519	0.01519	0.010127	0.010127
G	0.02661	0.05322	0	0.085793	0.064345	0.064345	0.042896	0.042896
L1	0.007595	0.01519	0.068071	0	0.007453	0.007453	0.004969	0.004969
L2	0.005696	0.011392	0.051054	0.007453	0	0.00559	0.003727	0.003727
L3	0.005696	0.011392	0.051054	0.007453	0.00559	0	0.003727	0.003727
L4	0.003797	0.007595	0.034036	0.004969	0.003727	0.003727	0	0.002484
L5	0.003797	0.007595	0.034036	0.004969	0.003727	0.003727	0.002484	0

As expected, the diagonal of the matrix has zero elements and the sum of all the elements in the matrix is 1.

6.7 Converting the PDF to a CDF

The probability density function (PDF) is then converter to a cumulative distribution function (CDF) by integration. The integration order is run along the row, then the second row, then the third row, etc. As expected, the first element of the CDF array is zero and the last two elements are equal to 1. This is in recognition of the fact that the diagonal elements in the PDF are always zeros.

	B2	B1	G	L1	L2	L3	L4	L5
B2	0	0.004348	0.034755	0.044882	0.052477	0.060072	0.065135	0.070198
B1	0.074546	0.074546	0.135361	0.155614	0.170804	0.185994	0.196121	0.206248
G	0.232858	0.286078	0.286078	0.371871	0.436216	0.500561	0.543457	0.586353
L1	0.593948	0.609138	0.677209	0.677209	0.684662	0.692115	0.697084	0.702053

L2	0.707749	0.719141	0.770195	0.777648	0.777648	0.783238	0.786965	0.790692
L3	0.796388	0.80778	0.858834	0.866287	0.871877	0.871877	0.875604	0.879331
L4	0.883128	0.890723	0.924759	0.929728	0.933455	0.937182	0.937182	0.939666
L5	0.943463	0.951058	0.985094	0.990063	0.99379	0.997517	1	1

6.8 Generating a number of passenger origin-destination pairs from the CDF

In this sub-section, five examples on the random sampling of passenger origin-destination pairs.

This is carried out by generating a random number uniformly distributed between 0 and 1. In the examples shown below, they are shown with 3 decimal places for simplicity. In each case, the number is to find the first number in the CDF that is larger than the random number generated. Once this is found, the row of the element represents the origin floor of the passenger origin-destination pair, and the column index represents the destination floor.

Five examples:

- Random number: 0.772; next *smallest* number in the table larger than this number is: 0.777648, which has a row index of L2 and a column index of L1. So, the origin–destination pair is L2 to L1.

- Random number: 0.591; next *smallest* number in the table larger than this number is: 0.593948, which has a row index of L1 and a column index of B2. So, the origin–destination pair is L1 to B2.

- Random number: 0.001; next *smallest* number in the table larger than this number is: 0.004348, which has a row index of B2 and a column index of B1. So, the origin–destination pair is B2 to B1.

- Random number: 0.998; next *smallest* number in the table larger than this number is: 1, which has a row index of L5 and a column index of L4. So, the origin–destination pair is L5 to L4.

- Random number: 0.862; next *smallest* number in the table larger than this number is: 0.866287, which has a row index of L3 and a column index of L1. So, the origin–destination pair is L3 to L1.

7. Verification

In order to check the correctness of the final origin destination matrix, it is possible to carry out a validation process as presented in the steps discussed below:

1. The OD matrix is set up with dimensions of N by N and set to 0.

2. The number of trials is set to a large number (e.g., 100,000 trials).

3. The trial numbers are split into four groups in proportion to the percentage mix of traffic. For example, if the traffic mix is 0.4:0.3:0.2:0.1 for incoming, outgoing, inter-floor and inter-entrance traffic, then 40,000 trials are set aside

for incoming traffic passenger, 30,000 for outgoing traffic passengers, 20,000 for inter-floor passengers and 10,000 for inter-entrance passengers.

4. Starting with the 40,000 passenger trials (representing incoming traffic), random sampling is carried out on the origin floor for the passenger by using $\mathbf{Pr_{arr}}$, and random sampling is carried out on the destination floor using $\mathbf{U_{nor}}$. This gives an origin-destination pair: the corresponding entry in the OD matrix is incremented. If the origin floor and destination floors are equal the result is discarded, and the process repeated.

5. For the 30,000 passenger trials (representing outgoing traffic), random sampling is carried out on the origin floor for the passenger by using $\mathbf{U_{nor}}$ and random sampling is carried out on the destination floor for the passenger by using $\mathbf{Pr_{arr}}$. This gives an origin-destination pair: the corresponding entry in the OD matrix is incremented. If the origin floor and destination floors are equal the result is discarded, and the process repeated.

6. For the 20,000 passenger trials (representing inter-floor traffic), random sampling is carried out on the origin floor for the passenger by using $\mathbf{U_{nor}}$ and random sampling is carried out on the destination floor for the passenger by using $\mathbf{U_{nor}}$. This gives an origin-destination pair: the corresponding entry in the OD matrix is incremented. If the origin floor and destination floors are equal the result is discarded, and the process repeated.

7. For the 10,000 passenger trials (representing inter-entrance traffic), random sampling is carried out on the origin floor for the passenger by using $\mathbf{Pr_{arr}}$ and random sampling is carried out on the destination floor for the passenger by using $\mathbf{Pr_{arr}}$. This gives an origin–destination pair: the corresponding entry in the OD matrix is incremented. If the origin floor and destination floors are equal the result is discarded, and the process repeated.

8. At the end of the 100,000 trials, the OD matrix is divided by the number of trials (100000). The result should match the OD final overall matrix derived as shown in sections 5 and 6.

The source code in MATLAB that can be used to create the probability density function array, convert it to a cumulative distribution function and then generate random passenger origin destination pairs is listed in Appendix A. It comprises the main module and three functions.

8. Conclusions

This paper has presented a systematic methodology for converting the user requirement specification into an origin-destination matrix.

It converts the traffic conditions (i.e., traffic mix) and the floor percentages (the floor percentage arrivals and the floor percentage populations) into an origin destination matrix. The conversion process takes into consideration that irrational passenger behaviour must be disallowed (e.g., one passenger's journey cannot originate and terminate at the same floor). Thus, the diagonal of the matrix must be zeroed.

The origin destination matrix is a compact concise form for expressing the passenger movements within the building. It has been used for generating random

passenger destinations for calculating the round-trip time using the Monte Carlo simulation method of within elevator traffic simulation software, an example of which is shown at the end of the paper.

Nomenclature

ic	percentage of incoming traffic under the traffic mix
if	the percentage of inter-floor traffic under the traffic mix
ie	the percentage of inter-entrance traffic under the traffic mix
N	the total number of floors in the building
og	the percentage of outgoing traffic under the traffic mix
$P_{arr}(i)$	the percentage arrival from the i^{th} floor
p_{ij}	the passenger transition probability from the i^{th} floor to the j^{th} floor
U	the total building population
U(i)	the building population on the i^{th} floor

A. Appendix A. Source Code in MATLAB

1. Generate Passenger Origin Destination Pairs Randomly

Author: Lutfi Al-Sharif Date: 27th April 2020 version: 1.0

```
% this MATLAB script generates passenger origin–destination pairs randomly
% based on three main parameters:
% the floor populations
% the entrance floors bias ratios
% the mix of traffic.
```

2. Set the Parameters

This building has 10 floors The floor index runs from 1 to 10. floors 1, 2 and 10 are entrance floors floor 10 is probably a restaurant.

```
% set the entrance bias
EB = [0.3 0.3 0 0 0 0 0 0 0 0.4];
% set the population on each floor
U = [016016016010010010010010 20];
% set the traffic mix as percentages adding up to 1
% [i/c o/g i/f i/e]
% incoming: outgloing: interfloor: interentrance
Traffic = [0.45 0.45 0.1 0];
```

```
% set the number of required passengers
P = 12;
```

3. shuffle the seed to set it to a specific value

```
% in order to get the same sequence every run, set the seed to a specific
% value (e.g., 0 in this case).
sd = 0; % value of the seed
rng(sd)

% in order to get a different sequence everytime the programme runs, shuffle
the seed
rng('shuffle');
```

4. Create the PDF (probability density function)

call Create_PDF to create the PDF array.

```
PDF = Create_PDF(EB, U, Traffic).

PDF =

Columns 1 through 7

       0    0.0229    0.0229    0.0229    0.0143    0.0143    0.0143
  0.0229         0    0.0258    0.0258    0.0161    0.0161    0.0161
  0.0229    0.0258         0    0.0029    0.0018    0.0018    0.0018
  0.0229    0.0258    0.0029         0    0.0018    0.0018    0.0018
  0.0143    0.0161    0.0018    0.0018         0    0.0011    0.0011
  0.0143    0.0161    0.0018    0.0018    0.0011         0    0.0011
  0.0143    0.0161    0.0018    0.0018    0.0011    0.0011         0
  0.0143    0.0161    0.0018    0.0018    0.0011    0.0011    0.0011
  0.0143    0.0161    0.0018    0.0018    0.0011    0.0011    0.0011
  0.0029    0.0337    0.0309    0.0309    0.0193    0.0193    0.0193

Columns 8 through 10

  0.0143    0.0143    0.0029
  0.0161    0.0161    0.0337
  0.0018    0.0018    0.0309
  0.0018    0.0018    0.0309
  0.0011    0.0011    0.0193
  0.0011    0.0011    0.0193
  0.0011    0.0011    0.0193
       0    0.0011    0.0193
  0.0011         0    0.0193
  0.0193    0.0193         0
```

5. Convert the PDF to a CDF

CDF: Cumulative Distribution Function The CDF is ideal to use for the random sampling in order to randomly generate passenger origin–destination pairs.

```
CDF = PDF2CDF(PDF).

CDF =

Columns 1 through 7

       0    0.0229    0.0458    0.0686    0.0829    0.0972    0.1115
  0.1659    0.1659    0.1917    0.2175    0.2337    0.2498    0.2659
  0.3548    0.3806    0.3806    0.3836    0.3854    0.3872    0.3891
  0.4465    0.4723    0.4752    0.4752    0.4771    0.4789    0.4807
  0.5296    0.5457    0.5475    0.5494    0.5494    0.5505    0.5517
  0.5875    0.6037    0.6055    0.6073    0.6085    0.6085    0.6096
  0.6455    0.6617    0.6635    0.6653    0.6665    0.6676    0.6676
  0.7035    0.7196    0.7215    0.7233    0.7245    0.7256    0.7267
  0.7615    0.7776    0.7795    0.7813    0.7824    0.7836    0.7847
  0.8080    0.8418    0.8726    0.9035    0.9228    0.9421    0.9614

Columns 8 through 10

  0.1258    0.1401    0.1430
  0.2821    0.2982    0.3319
  0.3909    0.3927    0.4236
  0.4826    0.4844    0.5153
  0.5528    0.5539    0.5732
  0.6108    0.6119    0.6312
  0.6688    0.6699    0.6892
  0.7267    0.7279    0.7472
  0.7859    0.7859    0.8052
  0.9807    1.0000    1.0000
```

6. Generate P passenger origin destination pairs using the CDF

call GenPass to generate P passenger origin–destination pairs usnig the CDF.

```
POD = GenPass(P,CDF)

% The resultant array POD is a two dimensional array that has dimensions of
% 2 rows by P columns. The first row contains the passenger origin floor;
% the second row contains the passenger destination floors.
% each column pertains to one passenger (a passenger origin–destination
% pair).
% the floor indexing runs from 1 to N.

% End.

POD =

     1    10     1    10     4     7     1     6     2     2    10     4
     7     8     4     3     8     2     9    10     3     8     9     2.
```

_____.

```
function [PDF] = Create_PDF(EB, U, Traffic).

%This module, Create_PDF compiles the probability density function (PDF)
%for generating passenger origin–destination pairs.

% Author: Lutfi Al-Sharif
% Date: 27th April 2020% this function creates a square array that is referred
to as a PDF
% PDF stands for probability density function (PDF).
% it is a square array that has dimensions of N rows by N columns
% where N is the total number of floors in the building.

% it is used in order to generate random passneger origin–destination-pairs.
% it depends on two important parameters: the relative percentages of the
% floor populations of the occupant floors, the relative entrance bias of
% entrance floors; and the mix of traffic (the percentage incoming traffic,
% outgoing traffic, interfloor traffic and interentrance traffic).
```

1. Background Theory and References

the methodology presented here in this function is based on a method of producing the origin destination matrix in the following papers.

```
% Al-Sharif L and Abu Alqumsan A M. An Integrated Framework for Elevator
% Traffic Design under General Traffic Conditions Using Origin Destination
% Matrices, Virtual Interval and the Monte Carlo Simulation Method.
% Building Services Engineering Research and Technology 2015; 36(6):
% 728–750.

%Al-Sharif L and Abu Alqumsan A M. Generating the Elevator
% Origin–Destination Matrix from the User Requirements Specification under
% General Traffic Conditions. Elevator Technology 2016; 21: 1–13.
% Proceedings of Elevcon 2016. Madrid/Spain: The International Association
% of Elevator Engineers.

%Al-Sharif L. Building the Origin–Destination
% Matrix under General Traffic Conditions and Using it to Generate
% Passenger Origin–Destination Pairs (METE XII). Lift Report 2016;
% 42(3):24–33.

% This paper introduces the methodology by which any floor could
% simultaneously be an entrance floor and an occupant floor

% Al-Sharif L. The Universal Origin–Destination-Matrix with Dual
% Designation Floors as Entrances and Occupant Floors. Lift Report 2018;
% 44(2):36–45.
```

2. extract the total number of floors and traffic mix from the variable Traffic

```
N = length(U);

% The variable Traffic is a one dimensional array, of size 1x4% the four
variables inside Traffic represent the ratio of incoming,
% outgoing, interfloor and interentrance respectively.
% they should all be less than or equal to 1% the sum of the four values must
add up to 1%
% the first element is the incoming traffic
% extract it and assign it to the variable i_c
i_c = Traffic(1);
% the second element is the outgoing traffic
% extract it and assign it to the variable o_g
o_g = Traffic(2);
% the third element variable is the interfloor traffic
% extract it and assign it to the variable i_f
i_f = Traffic(3);
% the fourth element in the Traffic Array element variable is the interfloor
traffic
% extract it and assign it to the variable i_f
i_e = Traffic(4);
Error using Create_PDF (line 49)
Not enough input arguments.
```

3. Generate the PDF

the next step is to prepare the probability density function (PDF) square array.

```
% find the total population U which is the sum of the individual floor
% populations
Utotal = sum(U);
% normalise it by dividing each element of the array by the total
% population.
% this will result in an array that has a total sum of 1
Unor = U/Utotal;
```

4. produce the first draft arrays for the four traffic modes

```
% the transpose of an array A is A'
% the tanspose is used here in order to produce a square array
% by multiplying the transpose of a row vector by another row vector of the
% same size.
% A square array is produced.

i_c_array = (EB'*Unor);
o_g_array = (Unor'*EB);
i_f_array = (Unor'*Unor);
i_e_array = (EB'*EB);
```

5. process them to make them ready for the final PDF

```
% zero the diagonals
% this assumes rational passenger behaviour whereby a passenger cannot
% travel from a floor back to the same floor.

for i = 1:N
   i_c_array(i,i) = 0;
   o_g_array(i,i) = 0;
   i_f_array(i,i) = 0;
   i_e_array(i,i) = 0;
end
% re-adjust to compensate for the loss of the value of the diagonal elements

% find the adjusting factors for all four arrays (which are the current sum
% of all the elements in the array (expected to be smaller than 1 due to
% teh fact that the diagonal was zeroed.

Mi_c = sum(sum(i_c_array)); % sum up all the elements of ic
Mo_g = sum(sum(o_g_array)); % sum up all the elements in og
Mi_f = sum(sum(i_f_array)); % sum up all the elements in if
Mi_e = sum(sum(i_e_array)); % sum up all the elements in ie

% then divide the array by the adjusting factor in order to restore the sum
% of its elements to 1
if Mi_c ~ = 0
i_c_array = i_c_array/Mi_c;
end

if Mo_g ~ = 0
o_g_array = o_g_array/Mo_g;
end

if Mi_f ~ = 0
i_f_array = i_f_array/Mi_f;
end

if Mi_e ~ = 0
i_e_array = i_e_array/Mi_e;
end

% Each of the four arrays now must sum up to 1.
% this can be checked by using the instruction sum(sum(A))
```

6. find the value of the PDF as the weighted sum of all the four traffic mode arrays

```
% now find the value fo the final PDF by doing a weight sum of the four
% arrays, multiplying each by its strength from the traffic array.

PDF = i_c*i_c_array+ o_g*o_g_array + i_f*i_f_array + i_e*i_e_array;

% the sum of all the element of the final PDF should also sum up to 1.
end
```

PDF2CDF.m.

Extract the number of floors from the dimensions of the input argument, PDF and set to N 1

Convert the PDF to a CDF 1

```
function [CDFfromPDF] = PDF2CDF(PDF).
%This function converts the PDF to a CDF
% Author: Lutfi Al-Sharif
% Date: 27th April 2020
```

1. Extract the number of floors from the dimensions of the input argument, PDF and set to N

```
N=size(PDF,1); % as this is a square matrix, it does not matter if we take the number of rows or the number of columns.
```

```
Error using PDF2CDF (line 7)
Not enough input arguments.
```

2. Convert the PDF to a CDF

the pdf is the probability density function. The cdf is the cumulative distribution function by integrating the PDF we can obtain the CDF.

```
CDFtemp = 0; % initialise a temporary variable that is hold a temporary value during the conversion process.
CDF(N, N) = 0; % initialise all the values in the CDF to zero

for i = 1:N % run through the array one row at a time, so finish a complete row then move to the next row and complete it.
% (i.e., rather than one column at a time).
for j = 1:N % move through column entres in the row being processed
    CDFtemp = CDFtemp+PDF(i,j); % keep track of the running sum in CDFtemp
    CDF(i,j) = CDFtemp;
end
end
% note that this CDF will always end with a 1, but might not start with a
% zero.
% it is important to be aware of this when carrying out random sampling on
% this CDF matrix.

CDFfromPDF=CDF;
end
```

GenPass.m.

Generate Passenger origin–destination pairs for P passengers 1

Set up a Counter and apply it to the CDF to find the origin–destination pair 1

```
function [ POD ] = GenPass(P,CDF)
```

1. Generate Passenger origin–destination pairs for P passengers

Author: Lutfi Al-Sharif Date: 27th April 2020

```
% this function will generate a number of passenger origin–destination
% pairs.
% The number of passengers is passed as a parameter P
% and the CDF (cumulative distribution function) is also passed to the
% function.
% it applies the principle of random sampling in order to find the
% origin–destination pair for a certain passenger.
% A random number that is uniformly distributed between 0 and 1 is
% generated using the function rand()
% this random number is then checked to see where it falls within the CDF
% array. Its position within the array decides the origin (the row) and
% the destination (the column) in the array.

% the output set of origin–destination pairs is compiled in the array
% called POD. it has dimensions of 2 x P.
% the first row contains the origins of the P passengers.
% the second row contains the correspinding destinations for these
% passengers.
```

2. Set up a Counter and apply it to the CDF to find the origin–destination pair

```
% the results will be placed in the POD with is a 2 by P array that
% contains all the passenger origins (in the first row) and their
% destinations (in the second row).
POD = 0; % zero the POD array.

% first set up a counter for the P passengers, using k as an index
for k = 1:P
% the ODfound is a Boolean variable that indicates that the value of
% the random variables has been matched to its position inside the CDF
% matrix.
% initialise it to false so that it can be set to true one found.
   ODfound = false; % the origin destination for this passenger has not been
found yet.
   tempCDF = 0; % tempCDF will hold that las indexed value of the elements in
the CDF.
   temp = rand();
%now run through the rows of the CDF one row at a time
for i = 1:length(CDF)
%now run through the elements of the row one column at a time
for j = 1:length(CDF)
% check if the random number is between two consecutive elements
% of the CDF array.
if ((temp>tempCDF) && (temp<=CDF(i,j)) && (ODfound == false))
            ODfound = true; % if found, then set temporary flag to indicate this
            POD(1,k) = i; % the row index is the origin for this kth passenger.
            POD(2,k) = j; % the column index is the destination for this kth
passenger
% the found origin and destination are placed in the kth column
% of the POD array. The origin in the first row and the
% destination in the second row.
```

```
end
end
end
end.

Error using GenPass (line 33)
Not enough input arguments.

end
```

Author details

Lutfi Al-Sharif
Mechatronics Engineering Department, The University of Jordan, Amman, Jordan

*Address all correspondence to: lutfi.alsharif@outlook.com

References

[1] Al-Sharif L, Al-Adem MD. The current practice of lift traffic design using calculation and simulation. Building Services Engineering Research and Technology. 2014;**35**(4):438-445

[2] Basagoiti R, Beamurgia M, Peters R, Kaczmarczyk S. Origin destination matrix estimation and prediction in vertical transportation. In: 2nd Symposium on Lift and Escalator Technologies. Northampton, U.K.: University of Northampton; 2012

[3] Basagoiti R, Beamurgia M, Peters R, Kaczmarczyk S. Passenger flow pattern learning based on trip counting in lift systems combined with on-line information. In: 3rd Symposium on Lift and Escalator Technologies. Northampton, U.K.: University of Northampton; 2013

[4] Kuusinen JM, Sorsa J, Siikonen ML. The elevator trip origin-destination matrix estimation problem. Transportation Science, articles in advance, INFORMS. 2014;**49**(3):433-719

[5] Kuusinen JM, Malapert A. The effect of randomisation on constraint based estimation of elevator trip origin-destination matrices. In: 4th Symposium on Lift and Escalator Technologies. Northampton, United Kingdom: University of Northampton; 2014

[6] Siikonen ML. Procedure for Controlling an Elevator Group Where Virtual Passenger Traffic Is Generated. US Patent Number 6 345 697 B1; 2002

[7] Kameli N. Predictor Elevator for Traffic During Peak Conditions. US Patent Number 5 276 295; 1994

[8] Al-Sharif L. Building the origin-destination matrix under general traffic conditions and using it to generate passenger origin-destination pairs (METE XII). Lift Report. 2016;**42**(3):24-33

[9] Al-Sharif L, Abu Alqumsan AM. An integrated framework for elevator traffic design under general traffic conditions using origin destination matrices, virtual interval and the Monte Carlo simulation method. Building Services Engineering Research and Technology. 2015;**36**(6):728-750

[10] Barney GC, Al-Sharif L. Elevator Traffic Handbook: Theory and Practice. 2nd ed. Routledge: Taylor & Francis, Abingdon-on-Thames, United Kingdom; 2016

[11] CIBSE. CIBSE Guide D: Transportation systems in buildings. Balham, London, United Kingdom: Chartered Institute of Building Services Engineers. 4th Edition 2010. pp. 2-4, Section 4.3.2

[12] British Council for Offices. BCO Guide to Specification 2009. London, United Kingdom: British Council for Offices; 2009

[13] Peters R. Lift traffic analysis: Formulae for the general case. Building Services Engineering Research and Technology. 1990;**11**(2):65-67

[14] Hakonen H, Lahdelma R. Calculation of Elevator Round-Trip Time for the Collective Control Algorithm in General Traffic Situations. Turku Centre for Computer Science, TUCS Technical Report No 671; 2005

[15] Al-Sharif L, Abu Alqumsan AM. Stepwise derivation and verification of a universal elevator round trip time formula for general traffic conditions. Building Services Engineering Research and Technology. 2015;**36**(3):311-330. DOI: 10.1177/0143624414542111

[16] Al-Sharif L. Calculating the elevator round trip time for the Most basic of cases (METE II). Lift Report 2014. 2014;**40**(5)

[17] Peters Research Ltd., Elevate. Available from: https://www.peters-research.com/index.php version 8.27