Final Report
February 2016

SAFE AND EFFICIENT PEDESTRIAN ACCOMODATION AT COORDINATED SIGNALIZED INTERSECTIONS

SOLARIS Consortium, Tier 1 University Transportation Center
Center for Advanced Transportation Education and Research
Department of Civil and Environmental Engineering
University of Nevada, Reno
Reno, NV 89557

Zong Tian, PhD, PE
Ali Gholami, PhD
Center for Advanced Transportation Education and Research
Department of Civil and Environmental Engineering
University of Nevada, Reno
Reno, NV 89557

DISCLAIMER

The contents of this report reflect the views of the authors, who are responsible for the facts and accuracy of the information presented herein. This document is disseminated under the sponsorship of the U.S. Department of Transportation's University Transportation Centers Program, in the interest of information exchange. The U.S. Government assumes no liability for the contents or use thereof.

EXECUTIVE SUMMARY

The purpose of this research is to study how pedestrian crossing timing should be considered in coordinated signal operations. A practical guideline needs to be developed to determine when accommodation (A) of pedestrian timing into coordination is preferable over non-accommodating (NA). With this guideline, practitioners input cycle length (C), volume (v), required pedestrian timing (RPT), and other signal parameters. The guideline will lead to a recommendation on whether A or NA should be used based on arterial vehicle delay. As part of the guideline development, a mathematical model was developed and validated by simulating 3,456 scenarios in VISSIM traffic simulation. Then, a software tool was created based on the mathematical model, named PeTASC (Pedestrian Timing Accommodation into Signal Coordination). A link is provided to download this free software. PeTASC can be used as a reference for an appropriate pedestrian timing design. This software can help practitioners design a better coordination plan and as a result, reduce arterial delay.

Keywords: Pedestrian timing accommodation; signal coordination; transition methods, PeTASC

ACKNOWLEDGMENTS

We wish to gratefully acknowledge the contributions of several people who made the progress of this research possible. This especially includes Dr. Ben Wang, Dr. Rasool Andalibian, Dr. Qi Xu, Mr. Mao Dean and Mr. Andrew Jayankura. Special thanks go to Mrs. Erika Hutton for her effort on proofreading the report.

GLOSSARY OF ACRONYMS

CATER: Center for Advanced Transportation Education and Research

GUI: Graphical User Interface

ITE: Institute of Transportation Engineers

NDOT: Nevada Department of Transportation

A: Accommodating Pedestrian Timing

NA: Not-accommodating Pedestrian Timing

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

INTRODUCTION

The majority of pedestrian accidents occur at signalized intersections. Increasing mobility and safety for pedestrians has been a major initiative, particularly in light of the recent federal focus on the Americans with Disability Act (ADA). The Manual on Uniform Traffic Control Devices (MUTCD) has adopted a new standard for determining pedestrian clearance times when crossing signalized intersections, where pedestrian walking speed is decreased from the previously adopted 4 feet per second to 3.5 feet per second. This revised standard will yield longer pedestrian crossing intervals, which can negatively affect signal system efficiency and increase congestion. Therefore, the intent of providing pedestrian safety may be compromised by increased driver frustration and vehicle collision hazards. The impact could be more dramatic for coordinated signal systems where a choice must be made among two pedestrian handling alternatives while developing coordinated signal timing plans: non-accommodating or accommodating pedestrian timing (e.g., using a longer cycle length). . These two alternatives affect coordinated signal systems in different ways. When pedestrian timing is accommodated, longer cycle length is generally needed. A longer cycle length results in longer delays under low volume conditions. On the other hand, when pedestrian timing is not accommodated due to use of a shorter cycle length, disruption to coordination can occur if a pedestrian crossing causes a signal to go into transition. In order to achieve optimal system performance, the conditions for when one alternative is preferred over another must be clearly identified, based on which guidelines can be developed for practicing signal engineers. Currently, such guidelines do not exist in published literature.

The primary objectives of this research are: (1) to address how pedestrian volume levels, pedestrian timing, signal transition methods, signal splits, number of intersections, weight of the side street (compared to the main street) and cycle length affect vehicle delay in a context of coordinated operation; and (2) to develop a guideline for selecting the appropriate pedestrian timing alternative for the best system performance. Due to the wide range of parameters affecting the results, this guideline is provided as a software.

LITERATURE REVIEW

Pedestrian crossing and handling alternatives at signalized intersections often involve conflicting objectives and need a balanced consideration of both safety and efficiency. Additionally, operational strategies should also target both pedestrian service quality and vehicular service quality. For coordinated signal systems, accommodating pedestrian timing in the signal timing plans could provide improved service to pedestrians while presenting negligible impacts on vehicular traffic, especially when longer signal cycle needs are driven by vehicular traffic demands. However, when vehicle and pedestrian volumes are low, accommodating pedestrians may require an unnecessary long cycle length, which can result in long delays and driver frustration. Under what conditions pedestrian timing should be accommodated and how it impacts the overall system efficiency has been a long debating subject among practicing traffic engineers. Discussions on this topic can be found in several documents (FHWA, 2008; Parsonson, 1992; Zegeer et al., 1982 and 1984; Petzold, 1997); however, limited guidelines are provided on how to select a pedestrian timing alternative under certain traffic conditions. Tian et al. (2001 and 2004) have conducted research on this topic; however, their findings were limited to some empirical evidence without robust methodologies being developed or comprehensively tested in the field. The real research issue lies in how different levels of pedestrian volumes affect the two pedestrian timing alternatives by considering a wide range of traffic volume and network conditions as well as the signal transition impacts on signal coordination.

For coordinated signal systems, traffic engineers have used two general strategies for timing signals to deal with pedestrian crossings: timing based on pedestrian minimums and timing based on vehicle minimums. Tian et al. (1999 and 2000) have analyzed the various effects of the two pedestrian treatment alternatives through case studies. It was found that although timing based on vehicle minimums can generally result in a shorter system cycle length, timing based on pedestrian minimums can normally achieve the same operational efficiency. The most significant advantage of timing based on pedestrian minimums is that the signal system will always remain in coordination. According to their study, the only drawback of timing based on pedestrian minimums was the likely use of longer cycle length. It was recommended that timing based on the pedestrian minimum technique should be applied when longer cycle length is required for the system and medium to high level pedestrian crossing activities exist.

Timing based on pedestrian minimums requires accommodation of pedestrian crossing time in the controller phase splits. The major advantage of this strategy is that the signal will remain in coordination regardless of whether there is a pedestrian phase activation; thus, it is a preferred alternative from the point of view of system operations (Tian and Urbanik, 2001). Tian et al. (2001) provided various alternatives for pedestrian timings under split-phasing operations. The advantages and disadvantages, implementation strategies, and potential impact on intersection operations, especially on coordinated signal systems, are addressed with regard to each timing alternative. In another study, Tian et al. (2006) proposed a model that consists of the probability of having a pedestrian call in a cycle, and the corresponding capacities and delays for the traffic movements. The model was compared with the SimTraffic simulation model based on a generic intersection with semi-actuated signal control, and was found to produce consistent delay results between the two models. Using the proposed model, the effects of pedestrians on intersection capacity and delay were analyzed. They concluded that the vehicle movement that is concurrent with pedestrian movement can achieve a higher capacity with a shorter cycle length with the same pedestrian volumes.

None of these studies, however, have dealt with the impact of accommodating pedestrian timing on coordinated signal systems in much detail. In the next section, a mathematical model will be developed that can be used as a robust method for the analysis of accommodating pedestrian timing under different volume and signal conditions.

MATHEMATICAL MODEL

Accommodating pedestrian timing in coordinated signal systems implies that the cycle length must be long enough to accommodate the Walk and Flash-Don't-Walk (called pedestrian clearance interval) for the side street. The advantage of this treatment is that a signal will always remain in coordination regardless of whether a pedestrian is crossing or not. The major disadvantage is the resulting longer cycle length, which may be operationally inefficient when vehicular traffic volume is low. Unnecessary long vehicle delays can result. The other alternative is to not accommodate pedestrian timing, i.e., the side street phase split is shorter than what is needed for pedestrian crossing. Although this alternative has no pedestrian safety degradation (i.e., pedestrians still receive the same WALK and Flash-Don't-Walk times during a crossing), the signal may go into transition, which will disrupt coordination, negatively affecting the system efficiency. In order to determine what pedestrian timing alternative is the best, this section proposes a mathematical model that relates vehicle volume, pedestrian volume, and network conditions to the delay caused by adding pedestrian timing.

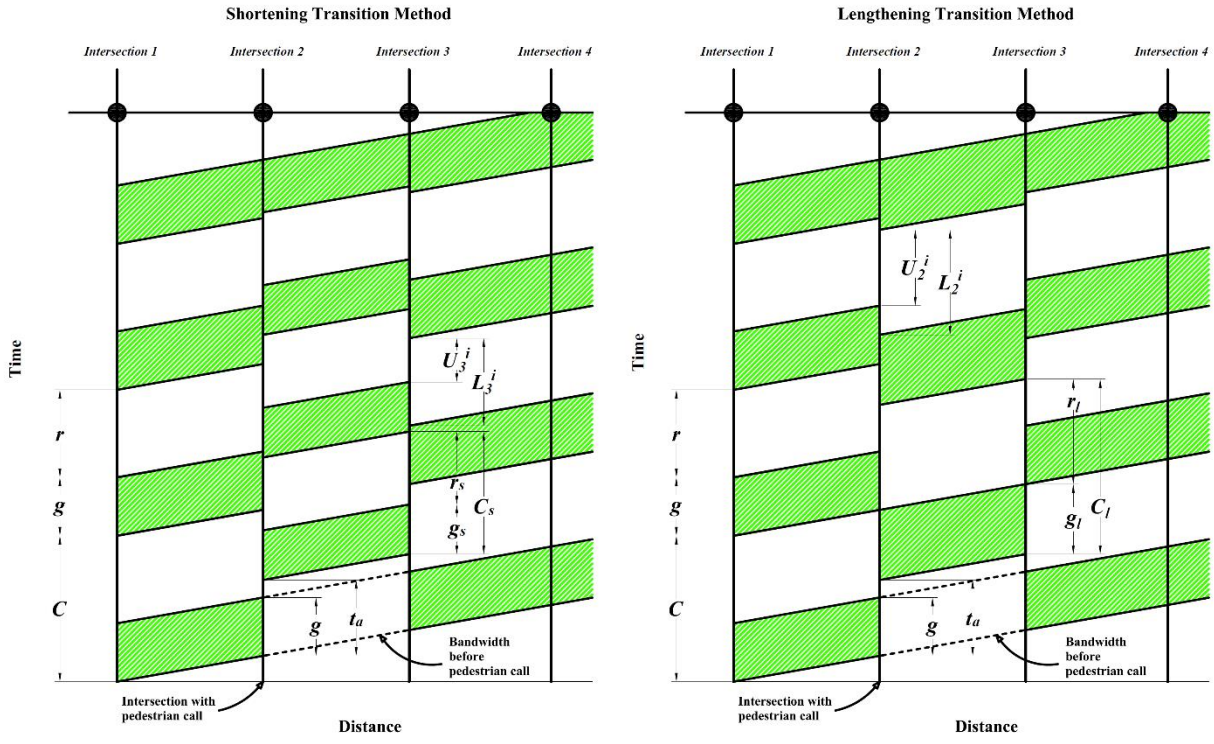

Figure 1: Transition due to non-accommodating pedestrian timing at coordinated signals

Non-accommodating PT

Suppose Intersections 1 to 4 in Fig. 1 are coordinated and pedestrian timing (PT) is not accommodated into traffic signal splits. If there is a pedestrian call at Intersection 2 to cross the main street and the required PT is more than the green time of the side street (for simplicity, green time through this section means green time plus yellow and all red), then green time of the main street (g) will be delayed by the additional time needed for pedestrians to cross the main street at the first cycle in transition. The value of additional time depends on the side street green time interval and required pedestrian time and can be calculated by the following equation:

$$t_a = t_p - g_s$$

Equation 1

t_a: Additional time needed for green time of the side street for pedestrians to cross main street (sec)

t_p: Required pedestrian time (walk time plus flash do not walk plus clearance time) (sec)

g_s: Side street green time (sec)

Theoretically, if $t_a \geq Cycle\ Length$, then only the remainder of the cycle length divided by t_a should be used as t_a. However, in practice this issue is very unlikely and therefore, it will not be discussed here.

As the next transition cycles, the amount of t_a depends on the transition method and the number of transition cycles during one transition period. The number of transition cycles during one transition period can be calculated by:

$$\beta_s = \left[\frac{t_a}{C \times \mu}\right]^+$$

Equation 2

$$\beta_l = \left[\frac{C - t_a}{C \times \mu}\right]^+$$

Equation 3

β_s: The number of transition cycles during one transition period at the shortening transition method. The plus in the equation means that the value should be rounded to the next integer.

β_l: The number of transition cycles during one transition period at the lengthening transition method. The plus in the equation means that the value should be rounded to the next integer.

C: Cycle length (sec)

μ: The maximum percentage of cycle length that is permissible to add to or to subtract from cycle length.

From equations 2 and 3, the amount that is subtracted from each cycle (shortening method) or added to each cycle (lengthening method) can be calculated by:

$$\alpha_s = -\frac{t_a}{\beta_s}$$

Equation 4

$$\alpha_l = \frac{C - t_a}{\beta_l}$$

Equation 5

α_s: The amount that is added to each cycle to lengthen the cycle length at the shortening method

α_l: The amount that is added to each cycle to lengthen the cycle length at the lengthening method

The time distance of the lower bound of delayed green to the beginning of the next green time at Intersections 2 and 3 can be calculated by the following equations:

$$L_2^i = \begin{cases} t_a + (i-1)\alpha & if \quad t_a + (i-1)\alpha \le r_t \\ r_t & if \quad t_a + (i-1)\alpha > r_t \end{cases}$$

$$i = 1, 2, \dots, \beta$$

Equation 6

$$L_3^i = \begin{cases} C - t_a - (i-1)\alpha & if \quad C - t_a - (i-1)\alpha \le r \\ r & if \quad C - t_a - (i-1)\alpha > r \end{cases}$$

$$i = 1, 2, \dots, \beta$$

Equation 7

L_2^i: The time distance of the lower bound of delayed green to the beginning of the next green time at Intersection 2 in Cycle *i* during transition

L_3^i: The time distance of the lower bound of delayed green to the beginning of the next green time at Intersection 3 in Cycle *i* during transition

r_t: Main street red time interval during transition period (sec)

And the time distance of the upper bound of transition green (g_t) to the beginning of green time of Intersection 2 can be calculated by the following equations:

$$U_2^i = \begin{cases} t_a + (i-1)\alpha - g & if \quad t_a + (i-1)\alpha > g \\ 0 & if \quad t_a + (i-1)\alpha \leq g \end{cases}$$

$$i = 1, 2, \dots, \beta$$

Equation 8

$$U_3^i = \begin{cases} C - t_a - (i-1)\alpha - g & if \quad C - t_a - (i-1)\alpha - g > g_t \\ 0 & if \quad C - t_a - (i-1)\alpha - g \leq g_t \end{cases}$$

$$i = 1, 2, \dots, \beta$$

Equation 9

U_2^i: The time distance of the upper bound of delayed green to the beginning of the next green time at Intersection 2 in Cycle *i* during transition

U_3^i: The time distance of the upper bound of delayed green to the beginning of the next green time at Intersection 3 in Cycle *i* during transition

The amount of green time that is delayed by transition is:

$$g_{d,2}^i = L_2^i - U_2^i$$

Equation 10

$$g_{d,3}^i = L_3^i - U_3^i$$

Equation 11

$g_{d,2}^i$: Amount of green time that is delayed by transition at Intersection 2 in Cycle *i* during transition

$g_{d,3}^{i}$: Amount of green time that is delayed by transition at Intersection 3 in Cycle i during transition

When signals are coordinated, except for the first intersection, a portion of vehicles arrives as a platoon to each intersection. The portion of green time that bunched vehicles use at the beginning of green time of the main street can be calculated by:

$$g_b = \begin{cases} \frac{r \times v_m}{f_s} & if \ \frac{r \times v_m}{f_s} < g \\ g & if \ \frac{r \times v_m}{f_s} \geq g \end{cases}$$

Equation 12

g_b : The interval at the beginning of green time of the main street that vehicles dispatch as bunched from the first intersection (sec).

r: Red time interval of main street (sec)

v_m: Main street volume (vps)

f_s : Saturated flow rate (vps)

g: Green time interval of main street (sec)

Vehicles will be dispatched randomly for the rest of the green time interval. The amount of green time that vehicles are dispatched randomly can be calculated as follows:

$$g_r = g - g_b$$

Equation 13

g_r : The interval toward the end of green time of the main street that vehicles dispatch randomly from the first intersection (sec).

The value of g_b and g_r change during transition. The value of them during each cycle of the transition period can be calculated by the following equations:

$$g_{b,2}^{i} = \begin{cases} g_b & if\ \ U_2^i = 0\ and\ g_b\ < g_{d,2}^i \\ g_{d,2}^i & if\ \ U_2^i = 0\ and\ g_b\ \geq g_{d,2}^i \\ g_b & if\ \ U_2^i > 0\ and\ L_2^i - U_2^i = g \\ 0 & if\ \ U_2^i > 0\ and\ L_2^i - U_2^i < g\ and\ g_r\ \geq g_{d,2}^i \\ g_{d,2}^i - g_r & if\ \ U_2^i > 0\ and\ L_2^i - U_2^i < g\ and\ g_r\ < g_{d,2}^i \end{cases}$$

Equation 14

$$g_{r,2}^{i} = \begin{cases} g_{d,2}^i - g_b & if\ \ U_2^i = 0\ and\ g_b\ < g_{d,2}^i \\ 0 & if\ \ U_2^i = 0\ and\ g_b\ \geq g_{d,2}^i \\ g_r & if\ \ U_2^i > 0\ and\ L_2^i - U_2^i = g \\ g_{d,2}^i & if\ \ U_2^i > 0\ and\ L_2^i - U_2^i < g\ and\ g_r\ \geq g_{d,2}^i \\ g_r & if\ \ U_2^i > 0\ and\ L_2^i - U_2^i < g\ and\ g_r\ < g_{d,2}^i \end{cases}$$

Equation 15

$$g_{b,3}^{i} = \begin{cases} g_b & if\ \ U_3^i = 0\ and\ g_b\ < g_{d,3}^i \\ g_{d,3}^i & if\ \ U_3^i = 0\ and\ g_b\ \geq g_{d,3}^i \\ g_b & if\ \ U_3^i > 0\ and\ L_3^i - U_3^i = g_t \\ 0 & if\ \ U_3^i > 0\ and\ L_3^i - U_3^i < g_t\ and\ g_r\ \geq g_{d,3}^i \\ g_{d,3}^i - g_r & if\ \ U_3^i > 0\ and\ L_3^i - U_3^i < g_t\ and\ g_r\ < g_{d,3}^i \end{cases}$$

Equation 16

$$g_{r,2}^{i} = \begin{cases} g_{d,3}^i - g_b & if\ \ U_3^i = 0\ and\ g_b\ < g_{d,3}^i \\ 0 & if\ \ U_3^i = 0\ and\ g_b\ \geq g_{d,3}^i \\ g_r & if\ \ U_3^i > 0\ and\ L_3^i - U_3^i = g_t \\ g_{d,3}^i & if\ \ U_3^i > 0\ and\ L_3^i - U_3^i < g_t\ and\ g_r\ \geq g_{d,3}^i \\ g_r & if\ \ U_3^i > 0\ and\ L_3^i - U_3^i < g_t\ and\ g_r\ < g_{d,3}^i \end{cases}$$

Equation 17

$g_{b,2}^{i}$: The time interval at the beginning of green time of the main street at Intersection 2 that vehicles dispatch as bunched from the first intersection in Cycle *i* during transition (sec).

$g_{r,2}^{i}$: The time interval at the beginning of green time of the main street at Intersection 2 that vehicles dispatch randomly from the first intersection in Cycle *i* during transition (sec).

$g_{b,3}^{i}$: The time interval at the beginning of green time of the main street at Intersection 3 that vehicles dispatch as bunched from Intersection 2 in Cycle *i* during transition (sec).

$g_{r,3}^{i}$: The time interval at the beginning of green time of the main street at Intersection 2 that vehicles dispatch as randomly from Intersection 2 in Cycle *i* during transition (sec).

The average delay that vehicles experience because of transition during one cycle can be calculated by:

$$Z_{n,2}^{i} = \left[g_{b,2}^{i} \times f_s \left(\frac{L_2^i + \left(U_2^i + g_{r,2}^i\right)}{2}\right)\right] + \left[g_{r,2}^{i} \times v_m \left(\frac{\left(L_2^i - g_{b,2}^i\right) + U_2^i}{2}\right)\right] - \left[v_{s,2} \times \rho \left(\frac{(t_a)^2}{2}\right)\right]$$

Equation 18

$$Z_{n,3}^{i} = \left[g_{b,3}^{i} \times f_s \left(\frac{L_3^i + \left(U_3^i + g_{r,3}^i\right)}{2}\right)\right] + \left[g_{r,3}^{i} \times v_{m,} \left(\frac{\left(L_3^i - g_{b,3}^i\right) + U_3^i}{2}\right)\right]$$

$Z_{n,2}^{i}$: Delay of cycle *i* due to pedestrian call at Intersection 2 (veh-sec)

v_s : Side street volume (vps)

ρ: The weight of side street volume compared to main street ($0 \leq \rho \leq 1$)

The third term of Equation 18, refers to the side street and is added only to the first cycle in the transition period. The factor ρ is added to the model to leverage the importance of the main street compared to side street(s). Many agencies prefer a good progression along the main street that sometimes imposes more delays on side street(s) vehicles. If ρ is equal to 1, then the importance of one vehicle on the main street is equal to the side street, and if ρ is equal to zero, it means vehicles at side streets are completely overlooked.

If the pedestrian volume is high, before a transition period ends, another transition period starts because of another pedestrian call. To consider overlapping transition periods, the following equation calculates the number of cycles that go into transition before another pedestrian call occurs:

$$\gamma = \begin{cases} \dfrac{1}{P(X \geq 1)} & if \quad \dfrac{1}{P(X \geq 1)} \leq \beta \\ \beta & if \quad \dfrac{1}{P(X \geq 1)} > \beta \end{cases}$$

Equation 19

γ: The number of cycles that go into transition before another pedestrian call occurs.

$P(X \geq 1)$: Probability of having at least one pedestrian call during one cycle

For example if $P(X \geq 1) = 0.2$ and $\beta = 6$ then $\gamma = 5$. This means that six cycles are needed for the signal to return to coordination but after five cycles, there would be another pedestrian call.

The probability of having at least one pedestrian call at a cycle can be expressed by the following equation:

$$P(X \geq 1) = 1 - P(X = 0) = 1 - e^{-(v_p\,/3600)C}$$

Equation 20

v_p : Pedestrian volume for crossing the main street (pph)

By putting together the previous equations, the following equations can calculate the delay for one transition period at the intersection with pedestrian call (Intersection 2) and its next intersection (Intersection 3):

$$Z_{n,2}^{t} = \left[\sum_{i=1}^{\gamma}\left[\left[g_{b,2}^{i} \times f_s\left(\frac{L_2^i + \left(U_2^i + g_{r,2}^i\right)}{2}\right)\right] + \left[g_{r,2}^{i} \times v_m\left(\frac{\left(L_2^i - g_{b,2}^i\right) + U_2^i}{2}\right)\right]\right]\right] - \left[v_{s,2} \times \rho\left(\frac{(t_a)^2}{2}\right)\right]$$

Equation 21

$$Z_{n,3}^{t} = \sum_{i=1}^{\gamma}\left[\left[g_{b,3}^{i} \times f_s\ \left(\frac{L_3^i + \left(U_3^i + g_{r,3}^i\right)}{2}\right)\right] + \left[g_{r,3}^{i} \times v_{m,}\left(\frac{\left(L_3^i - g_{b,3}^i\right) + U_3^i}{2}\right)\right]\right]$$

Equation 22

$Z_{n,2}^{t}$: Delay of one transition period due to non-accommodating PT at Intersection 2 if pedestrian call occurs at Intersection 2

$Z_{n,3}^{t}$: Delay of one transition period due to non-accommodating PT at Intersection 3 if pedestrian call occurs at Intersection 2

The delay Z does not occur at all cycles. Therefore, to calculate a transition-caused delay for one hour, it should be determined how many transition periods happen during one hour. If

the increase and decrease of a cycle length during transition is overlooked, the number of transition periods can be calculated by:

$$N = \begin{cases} \dfrac{3600}{C \times \gamma} & if \ \dfrac{1}{P(X \geq 1)} \leq \beta \\ \dfrac{3600}{C} \times P(X \geq 1) & if \ \dfrac{1}{P(X \geq 1)} > \beta \end{cases}$$

Equation 23

N: Number of transition periods per hour

The length of the first cycle in transition is equal to $C + t_a$ and the length of the next cycles in transition for shortening and lengthening methods are calculated by the following equations:

$$C_t = C + \alpha$$

Equation 24

C_t: The length of cycles after the first cycle in transition period (sec)

Therefore, the average of cycle length during one hour can be calculated by:

$$C_a = \begin{cases} \dfrac{(C + t_a) + [(\gamma - 1)C_t]}{\gamma} & if \ \dfrac{1}{P(X \geq 1)} \leq \beta \\ \dfrac{(C + t_a) + [(\beta - 1)C_t]}{\beta} P(X \geq 1)\beta + C\big[1 - [P(X \geq 1)\beta]\big] & if \ \dfrac{1}{P(X \geq 1)} > \beta \end{cases}$$

Equation 25

C_a: Average cycle length during one hour (sec)

Using the value of average cycle length, Equation 23 can be rewritten as follow:

$$N = \begin{cases} \dfrac{3600}{C_a \times \gamma} & if \ \dfrac{1}{P(X \geq 1)} \leq \beta \\ \dfrac{3600}{C_a} \times P(X \geq 1) & if \ \dfrac{1}{P(X \geq 1)} > \beta \end{cases}$$

Equation 26

By multiplying Equations 21 and 22 to Equation 26, hourly delay of non-accommodating PT due to transition at Intersection 2 and 3 if pedestrian call occurs at Intersection 2 can be calculated as follow:

$$D_{n,2}^{h} = Z_{n,2}^{t} \times N$$

Equation 27

$$D_{n,3}^{h} = Z_{n,3}^{t} \times N$$

Equation 28

$D_{n,2}^{h}$: Hourly delay of non-accommodating PT at Intersection 2 if pedestrian call occurs at Intersection 2.

$D_{n,3}^{h}$: Hourly delay of non-accommodating PT at Intersection 3 if pedestrian call occurs at Intersection 2.

Equations 27 and 28 define the delay caused by non-accommodating PT at one intersection in the middle of a series of coordinated intersections. It does not consider the other delays imposed on vehicles.

If the pedestrian call occurs at the first intersection, the vehicles arriving at this intersection cannot be considered as bunched anymore because its previous intersection is not coordinated with it. As a result, the cycle length of this intersection compared to its previous intersection is usually different. Therefore, even if vehicles approach as a platoon because of another signalized intersection, this bunched arrival can be considered as random. The delay caused by pedestrian call and transition at this intersection can be modeled as follows:

$$Z_{n,1}^{t} = \left[\sum_{i=1}^{\gamma}\left[g_{d,1}^{i} \times v_m\left(\frac{L_1^i - U_1^i}{2}\right)\right]\right] - \left[v_{s,1} \times \rho\left(\frac{(t_a)^2}{2}\right)\right]$$

Equation 29

$Z_{n,2}^{t}$: Delay of one transition period due to non-accommodating PT at Intersection 1 if pedestrian call occurs at Intersection 1

All parameters of this equation can be calculated similarly to equations related to Intersection 2. Also, the delay of its next intersection can be calculated similarly to the delay of Intersection 3 when pedestrian call occurs at Intersection 2. Therefore, hourly delay of non-accommodating PT at Intersection 1 and 2 if pedestrian call occurs at Intersection 1 can be calculated as follows:

$$D_{n,1}^{h} = Z_{n,1}^{t} \times N$$

Equation 30

$$D_{n,2}^{h} = Z_{n,2}^{t} \times N$$

Equation 31

$D_{n,1}^{h}$: Hourly delay of non-accommodating PT at Intersection 1 if pedestrian call occurs at Intersection 1.

$D_{n,2}^{h}$: Hourly delay of non-accommodating PT at Intersection 2 if pedestrian call occurs at Intersection 1.

By summing up the total hourly delay of each intersection, the total delay of non-accommodating PT at intersection with pedestrian call and its next intersection can be calculated as follows:

$$D_{n,t}^{h} = \sum_{i=p}^{p+1} D_{n,i}^{h}$$

Equation 32

$D_{n,t}^{h}$: Total hourly delay caused by transition when PT is not accommodated at Intersection p (intersection with pedestrian call) and its next intersection.

p: The intersection number with pedestrian call

Accommodating PT

Figure 2 demonstrates the problem of adding t_a to side street green time. For the first intersection, the following equation can show the delay caused by accommodating PT into signal timing:

$$D_{a,1}^{h} = \left[\frac{(t_a)^2\left(V_m - V_{s,1}\right)}{2}\right]\left(\frac{3600}{C}\right) = \frac{1800\ (t_a)^2\left(V_m - V_{s,1}\right)}{C}$$

Equation 33

$D_{a,1}^{h}$: Delay caused by adding t_a to side street green time at Intersection 1 (veh-sec).

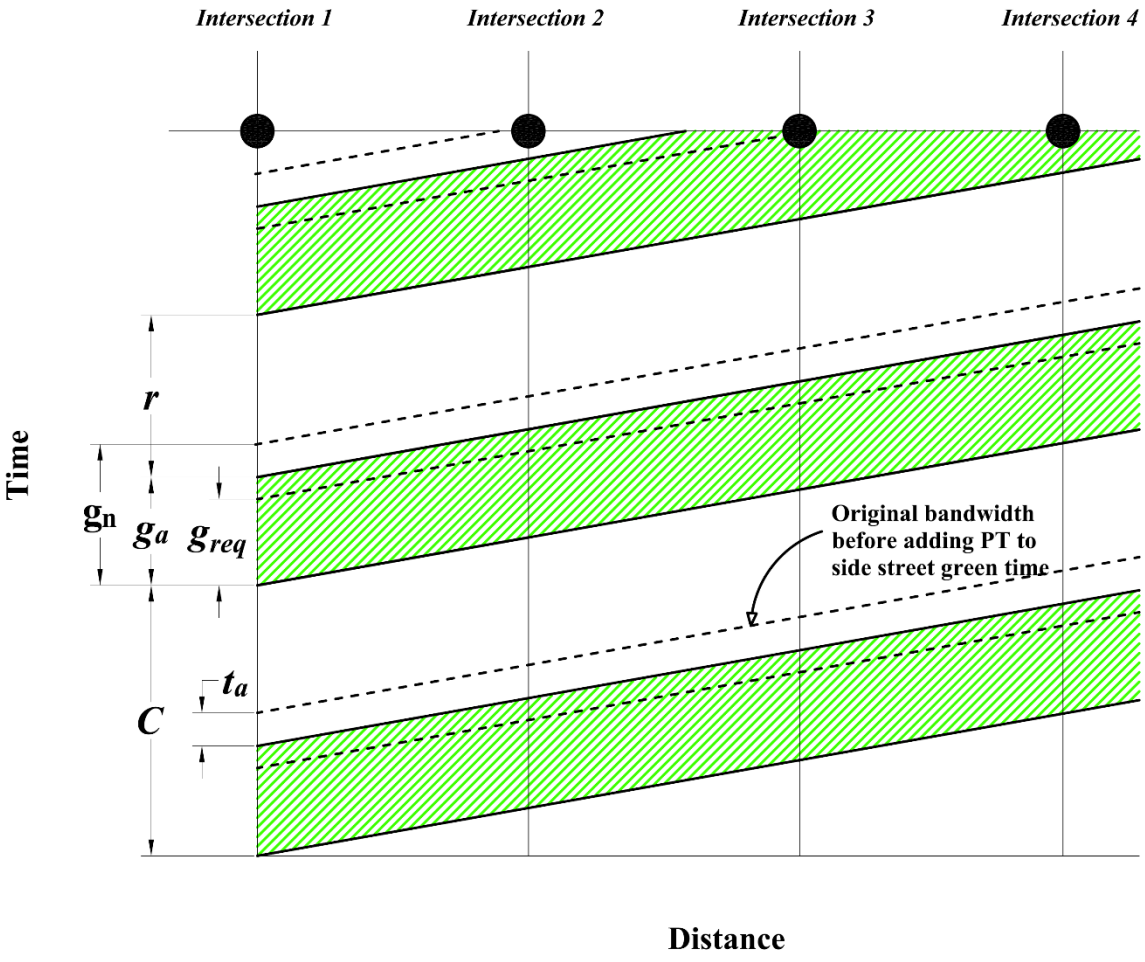

Figure 2: Accommodating pedestrian timing at coordinated signals

For the next intersections, if $required\ green(g_{req}) \leq accommodated\ green\ (g_a)$, then there is no delay at the main street caused by adding t_a to side street green time. However if $g_{req} > g_a$, then delay can be calculated by multiplying the number of vehicles that could not be served due to increasing the side street green time. Therefore, for the next intersections (Intersection 2, 3, …, n) the delay of adding t_a can be formulated as follows:

$$D_{a,i}^{h} = \begin{cases} \dfrac{1800\ (\ t_a)^2(-\rho \times v_{s,i})}{C} & if \quad g_{req} \leq g_a \\ \dfrac{1800\ (\min(t_a, g_{req} - g_a))^2(f_s - \rho \times v_{s,i})}{C} & if \quad g_{req} > g_a \end{cases}$$

$$i = 2, 3, \dots, n$$

Equation 34

$D_{a,i}^{h}$: Delay of accommodating t_a (adding t_a to side street green time) at Intersection i (veh-sec)

n: Number of coordinated signals

$v_{s,i}$: Side street *i* volume (vps)

Required green (g_{req}) can be calculated using the following equation:

$$g_{req} = \frac{v_m \times C}{f_s}$$

Equation 35

g_{req}: Required green based on main street volume and its saturation flow rate.

And accommodated green (g_a) can be calculated by:

$$g_a = g - t_a$$

Equation 36

g_a: Accommodated green (sec)

Note that in Equation 36, having an equal base in comparing the accommodating and non-accommodating PT, cycle length is considered equal for both methods; therefore, the additional time that is added to side street green time will be reduced from the main street green time. As a result, the value of t_a must not exceed the main street green time. Practically for a certain cycle length, maximum t_a can be determined by the following equation:

$$t_{a,max,a} \approx g - g_{req}$$

Equation 37

$t_{a,max,n}$: Maximum t_a for accommodating PT method

For example, suppose that the cycle length at an intersection is 60 seconds, the green time of the main street is 30 seconds, left turn green time of the main street is 10 seconds, and the required pedestrian time (Walk+ FDW+ Clearance) is 50 seconds. This means 30 seconds must be reduced from the main street green times (left turn and through). Therefore, this cycle length cannot accommodate pedestrian timing and needs to be increased to a bigger value.

Summing up Equations 33 and 34, the total hourly delay of accommodating PT can be calculated by:

$$D_{a,t}^{h} = \sum_{i=1}^{n} D_{a,i}^{h}$$

Equation 38

The effect of semi-actuated coordination

Equations 31 and 34 do not account for early green of the main street due to left turn or side street force-off on delay calculations. In other words, it assumes that coordination is running under fixed split timing. If coordination is semi-actuated, it is possible that the signal does not go into transition or t_a is reduced because of early green of the main street. The following sections investigate this issue for accommodating and non-accommodating methods.

1) Accommodating

Two extremes for this method exist. The first is when there is no vehicle at the side street and no pedestrian volume crossing the major street. In this situation, the main street remains green and therefore t_a is zero. The second is when intersections are saturated and $P(X \geq 1) = 1$. In this situation, t_a is always added to the side street green time. At other times, the amount of t_a is a function of side street volume and pedestrian volume:

$$t_a^a = f(v_s, v_p)$$

Equation 39

t_a^a: Modified t_a for accommodating PT at semi-actuated coordination.

For Equation 39, if there is even one pedestrian at each cycle, t_a would be at its maximum no matter what v_s is, but if there is no pedestrian, t_a remains zero assuming that the controller is able to terminate the side street green time at its original value. With this assumption, Equation 39 can be simplified as follow:

$$t_a^a = t_a \times P(X \geq 1)$$

Equation 40

2) Non-accommodating

When PT is not accommodated, t_a can be less than required t_a or even zero due to the main street left turn force-off. Therefore, the value of t_a depends on the volume of the main street left turn and the maximum volume that this left turn can serve per cycle. The left turn per cycle can be calculated by the following equation:

$$v_{l,c} = v_l \times C$$

Equation 41

$v_{l,c}$: Main street left turn volume per cycle

v_l: Main street left turn volume (vps)

The maximum volume that this left turn can serve per cycle can be calculated as follows:

$$c_{l,c} = f_s \times g_l$$

Equation 42

$c_{l,c}$: Capacity of left turn (vehicle per cycle)

g_l: Left turn green time interval (sec)

Based on Equations 41 and 42, the modified t_a for non-accommodating PT at semi-actuated coordination signals can be calculated by the following equation:

$$t_a^n = \begin{cases} t_a & if \quad \frac{v_{l,c}}{c_{l,c}} \geq 1 \ or \ \frac{v_{l,c}}{c_{l,c}} g_l + E \geq g_l \\ t_a - \left[\min\left(t_a, g_l - \left(\frac{v_{l,c}}{c_{l,c}} g_l + E\right)\right)\right] & if \quad 0 < \frac{v_{l,c}}{c_{l,c}} < 1 \ and \ \frac{v_{l,c}}{c_{l,c}} g_l + E < g_l \\ t_a - (\min(g_l, t_a) & if \quad v_l = 0 \end{cases}$$

Equation 43

t_a^a: Modified t_a for non-accommodating PT at semi-actuated coordination

E: Gap out extension (sec)

Note that t_a^a will not be used for the side street. Also, if the main street is one way, or if there is no phase for left turn, there is no early green because of left turn gap out and therefore, Equation 43 will not be applied on t_a.

The optimum method

For any situation, by calculating Equations 27 and 30, the two methods of accommodating and non-accommodating PT can be compared. The optimum method can be determined by the following equation:

$$A = \left[\frac{D_{a,t}^{h} - D_{n,t}^{h}}{D_{a,t}^{h}}\right] \times 100$$

Equation 44

A: Percentage of delay increase after accommodating PT

Therefore, if $A \gg 0$, non-accommodating PT is preferable. If $A \ll 0$, accommodating of PT is preferable; and if $A \cong 0$, then there is not a significant benefit of one method over the other one.

MODEL VALIDATION

A code was written in COM interface of VISSIM (appendix A) to model all combinations of Table 1. These combinations formed 3,456 scenarios. For each scenario, two methods of accommodation and non-accommodation of pedestrian timing were performed and delay increase of accommodating PT over non-accommodating was recorded. If the result is negative, it means PT accommodation is preferable. If the result is positive, non-accommodation is preferable. Figure 3 shows a sample of VISSIM simulation output. The outputs of these scenarios were compared with the ones obtained from the mathematical model.

Table 1: Range of simulation parameters

PARAMETER	FROM	INTERVAL	TO	NO OF CASES
Volume	50	50	1200	24
Pedestrian Volume	5	5	30	6
t_a	5	5	30	6
Cycle Length	60	20	120*	4

* for cycle lengths more than 120 sec, side street green time is long enough to accommodate pedestrian timing

Figures 3 and 4 compare one sample of the simulation and mathematical models. It can be seen that up to 500 vphpl, both the simulation and mathematical models do not show a significant difference between A and NA (though NA is slightly better in both models). After 500 vphpl, both models show that NA is significantly better than A. However, in the mathematical model, NA is only better when the transition method is shortening. In the simulation model, the "best" transition method was selected. For other scenarios, this comparison shows that the simulation results are close to the mathematical model. However, it can be noted that the simulation results fluctuate slightly from one volume to its adjacent volume. For example, in Figure 3, when volume is 200 vphpl and pedestrian volume is 30 pph, NA is slightly better, but when increasing the volume to 250 vphpl, NA and A are almost equal. For the comparison of simulation and mathematical models, these fluctuations are ignored.

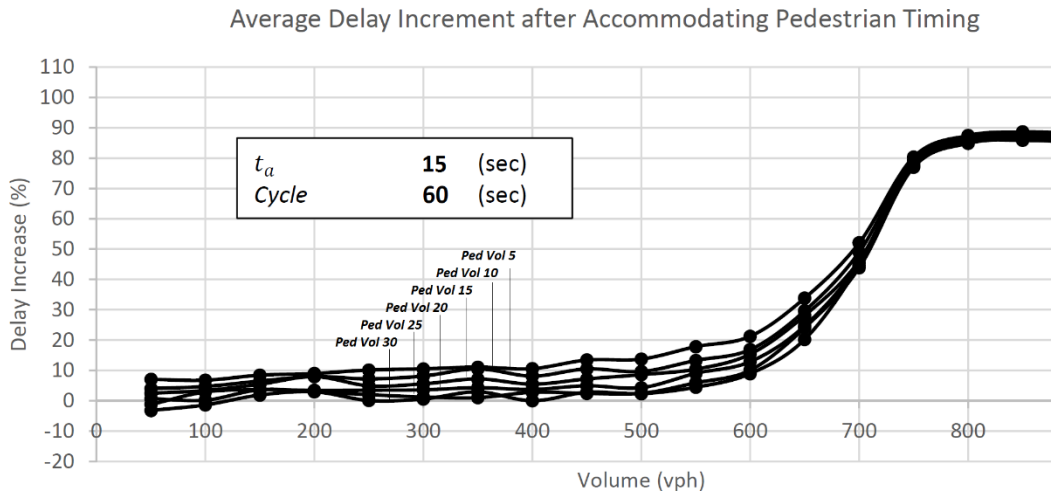

Figure 3: Simulation average delay increment after accommodating pedestrian timing, cycle 60 sec, $t_a = 15$ sec

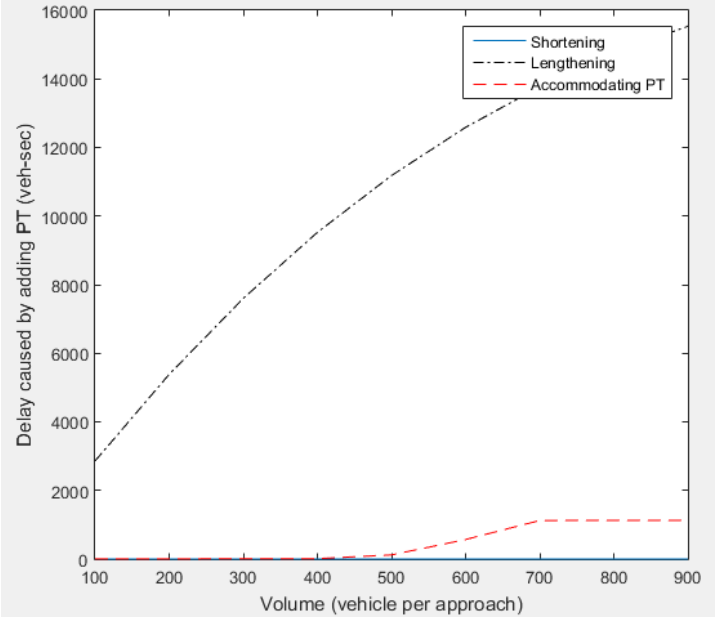

Figure 4: Mathematical model delay caused by adding pedestrian timing, cycle 60 sec, $t_a = 15$ sec, Ped Vol 30 pph

The distance of intersections in these scenarios was half a mile from each other. To check if the distance of intersections can affect the results, all 3,456 scenarios were retested by multiplying the distance of intersections by factors 0.5 and 1.5. The results of these three sets of scenarios were compared. Figure 5 shows a sample of the results. Figure 6 depicts a sample of the correlation of two different distance values. From these two figures, it can be seen that the results of each set is close to other sets. Similar analysis on other results shows that the distance of intersections does not affect the results significantly.

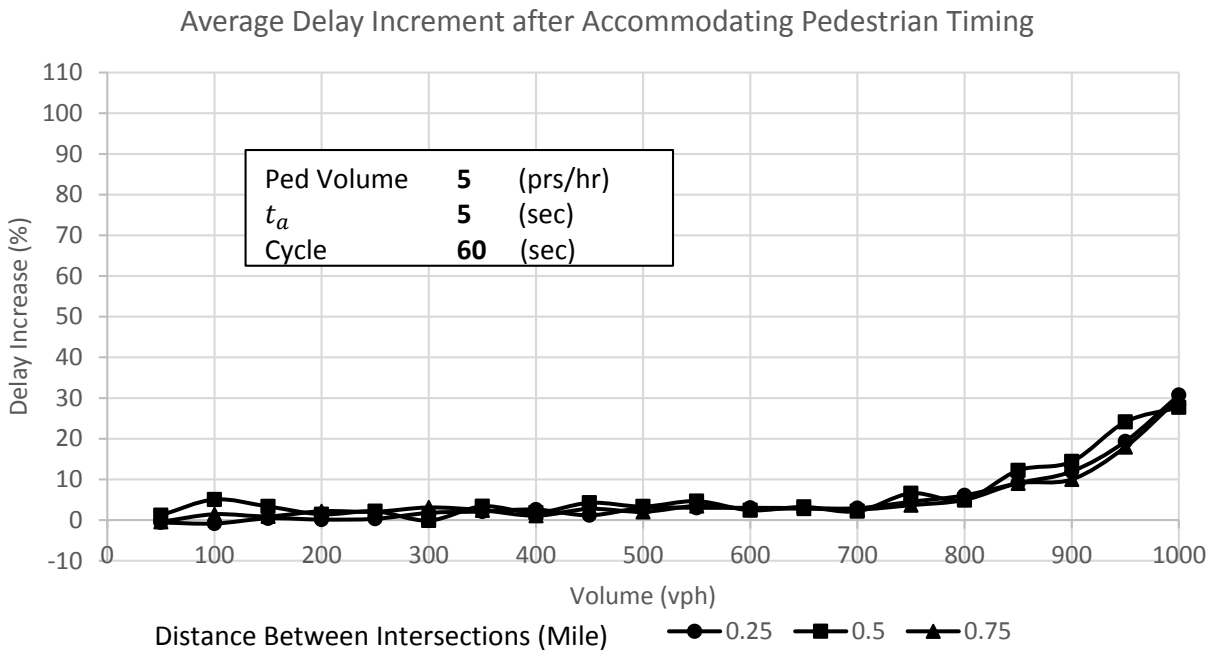

Figure 5: Sensitivity Analysis on distance between intersections

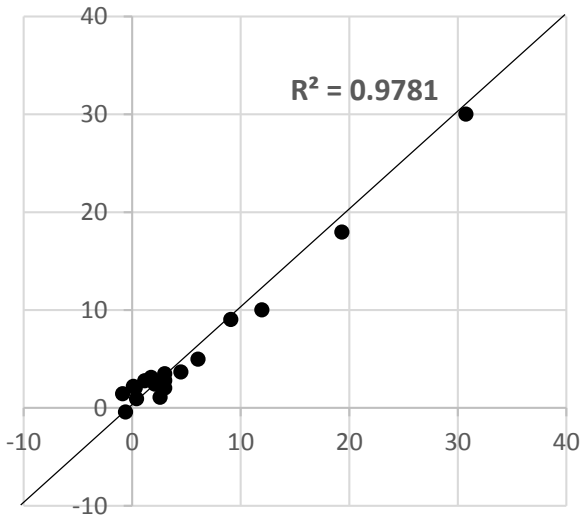

Figure 6: Correlation of 0.25 and 0.75 mile for the conditions of Figure 5

MODEL SOFTWARE

In this section, software is provided to select the best pedestrian accommodation method. Table 2 shows the inputs required for Equation 44. The design parameters are shown by bold font and can vary, thus changing the results. For example, the cycle length can affect the probability of having pedestrian(s) during a cycle, green time intervals, and optimum transition method. Finding the best value of each of these parameters is very time consuming for practitioners. To facilitate this procedure, a software has been developed based on the mathematical model called PeTASC (Pedestrian Timing Accommodation into Signal Coordination) that can be accessed by scanning the QR code or following the link in Figure 7. Its MATLAB code can be seen in Appendix B. Figure 8 shows a screen shot of PeTASC. With this software, a user can enter inputs and choose reasonable parameters for coordinated signals. Most of these design parameters can be designed in other software (such as Synchro or Vistro) including optimum cycle length and splits. However, parameters ρ and μ, as well as the best transition method and the best method for accommodating PT. To narrow down the design range, the transition method can also be removed from the design parameters because many controllers are now able to choose the optimum transition method, also called "best way" (also known as smooth, short way or fast way). If Table 3 is considered as the range of the remaining design parameters, by calculating 60 cases using PeTASC interface, the best one can be chosen among them. Figure 9 shows a sample of outputs for choosing the best ρ, μ, and PT accommodation method for given volume and signal parameters. This analysis indicates that for μ from 0.05 to 0.15, accommodating PT is preferable over both methods of transition. However, if a bigger amount of μ is selected, the shortening transition method reduces delays more so than lengthening and accommodating PT. PeTASC can analyze methods over other parameters, especially over volume. In other words, it can determine the volume boundaries that each method can aptly apply to PT. Figure 10 shows a sample output for given conditions.

https://drive.google.com/open?id=0B4juw5AdxVsYTzN5c3hnVUJmSnc

Figure 7: QR code and link address of PeTASC for download

Table 2: Model inputs

VARIABLE		SYMBOL	DESIGN PARAMETER
Volume	Saturated flow rate (vps)	f_s	No
	Main street volume (vps)	v_m	No
	Main street left turn volume (vps)	v_l	No
	Side street *i* volume (vps)	$v_{s,i}$	No
	Pedestrian volume for crossing main street (pph)	v_p	No
	The weight of side street volume compared to main street ($0 \leq \rho \leq 1$)	$\boldsymbol{\rho}$	**Yes**
Signal timing	**Cycle length (sec)**	$\boldsymbol{C}$	**Yes**
	Main street green time interval of (sec)	$\boldsymbol{g}$	**Yes**
	Left turn green time interval (sec)	$\boldsymbol{g_l}$	**Yes**
	Side street green time interval (sec)	$\boldsymbol{g_s}$	**Yes**
	Gap out extension (sec)	$\boldsymbol{E}$	**Yes**
	No. of intersection with pedestrian call	p	No
	Required pedestrian time (walk time plus flash do not walk plus clearance time) (sec)	t_p	No
	The maximum percentage of cycle length that is permit able to add to or to subtract from cycle length.	$\boldsymbol{\mu}$	**Yes**
	Number of coordinated signals	n	Could be

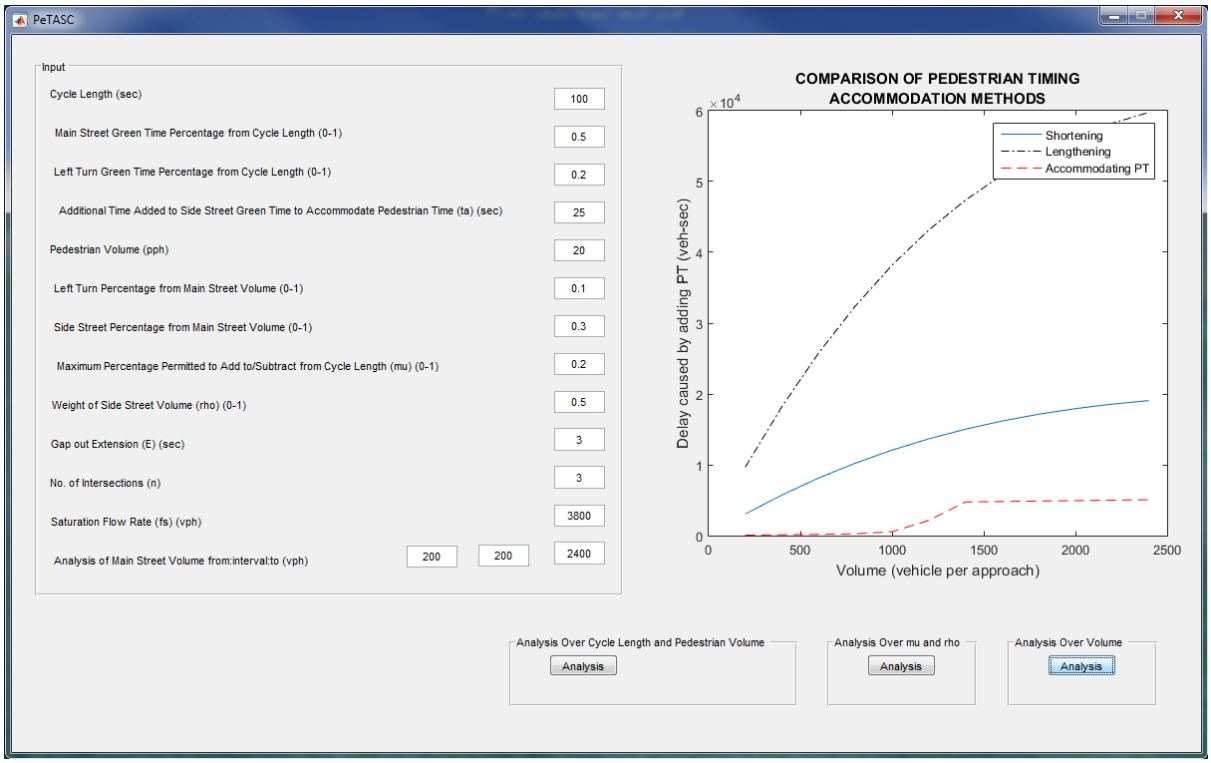

Figure 8: Screen shot of PeTASC

Table 3: Range of parameters

DESIGN PARAMETER	RANGE	INTERVALS	NUMBER OF CASES
ρ	$(0 \leq \rho \leq 1)$	0.25	5
μ	$(0.05 \leq \mu \leq 0.3)$	0.05	6
PT accommodation method	Yes / No	-	2

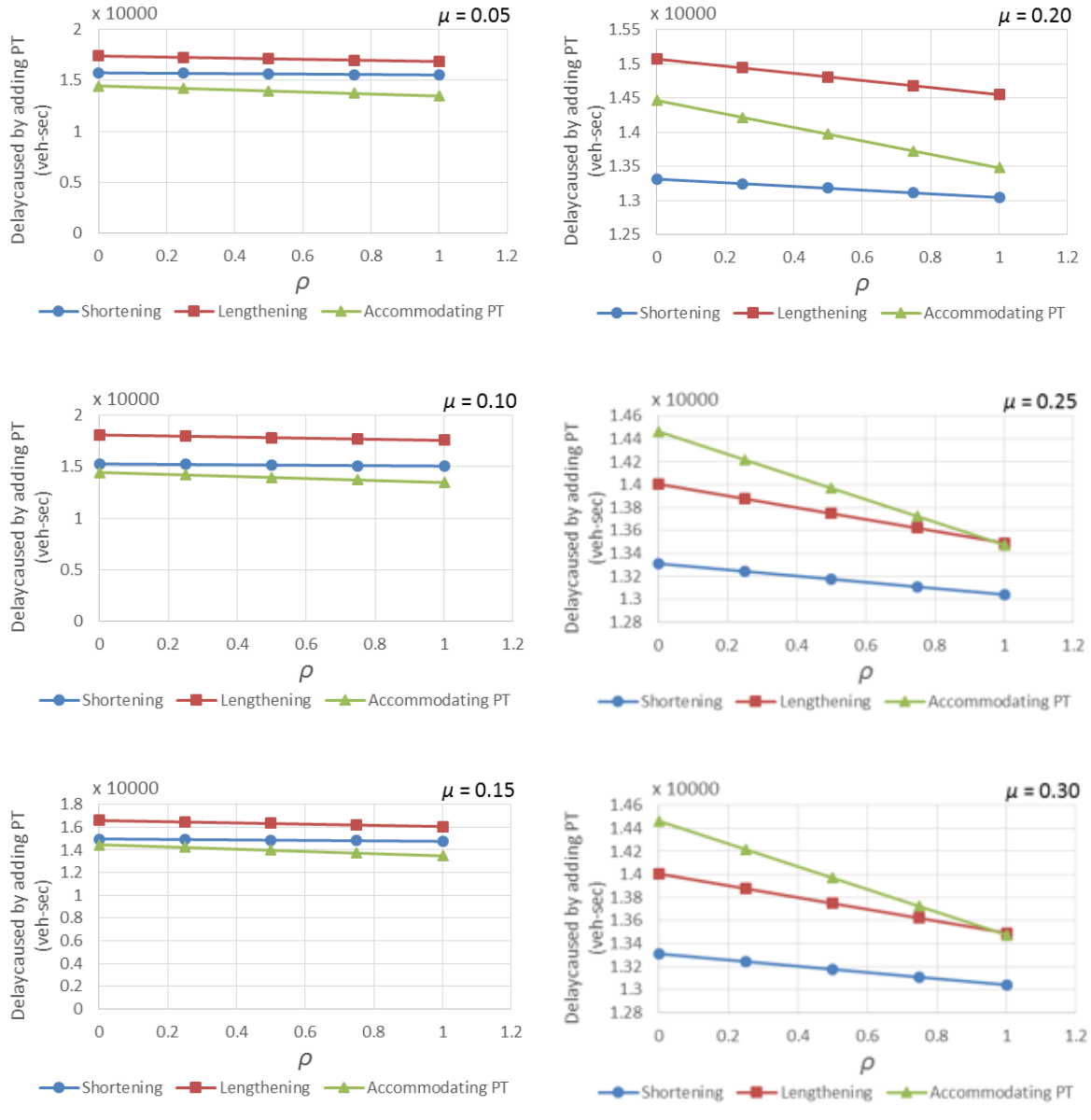

Figure 9: delay caused by adding PT to signal coordination, C: 80 sec; g: 0.5 C; g_l: 0.2 C; t_a: 45 sec; E: 3 sec; n: 3; v_m: 1200 vph; v_s: 0.3 v_m; f_s: 1.056 vps; v_l: 0.1 v_m; v_p: 45 pph

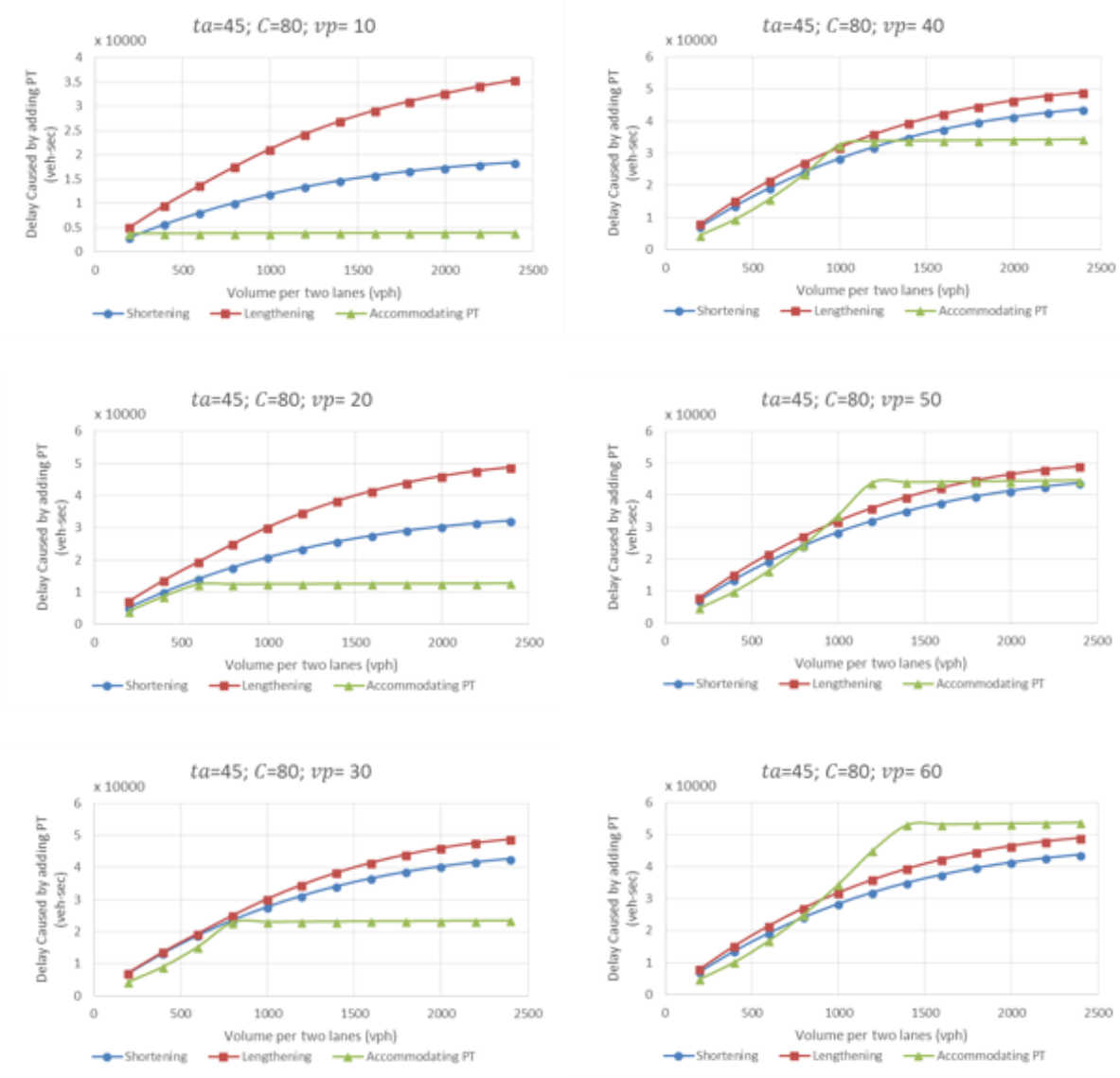

Figure 10: delay caused by adding PT to signal coordination, $\mu = 0.2$, $\rho = 1$

Figures 11 to 13 are samples of PeTASC outputs, which show average delay increments after accommodating pedestrian timing for different additional pedestrian time, cycle length and volume. These graphs were made based on the mathematical model and can be a reference for a wide range of conditions. For example, referring to Figure 11, the threshold for accommodating pedestrian timing is almost 950 vph for all pedestrian volume levels. This means that when additional pedestrian time (t_a) is 35 seconds and cycle length is 160 seconds, if volume is more than 950 vph, accommodation of t_a is more beneficial for this condition. The results are very sensitive to the amount of t_a. For example, Figure 12 diagrams are referring to the exact condition of Figure 11 except t_a is 20 seconds more. At this condition for all vehicle and pedestrian volume levels, accommodating t_a is more preferable compared to

non-accommodating. Figure 13 shows another example of sensitivity of results to t_a. In this figure, t_a is equal to 25 seconds and for all vehicle and pedestrian volume levels, non-accommodating pedestrian timing is preferable.

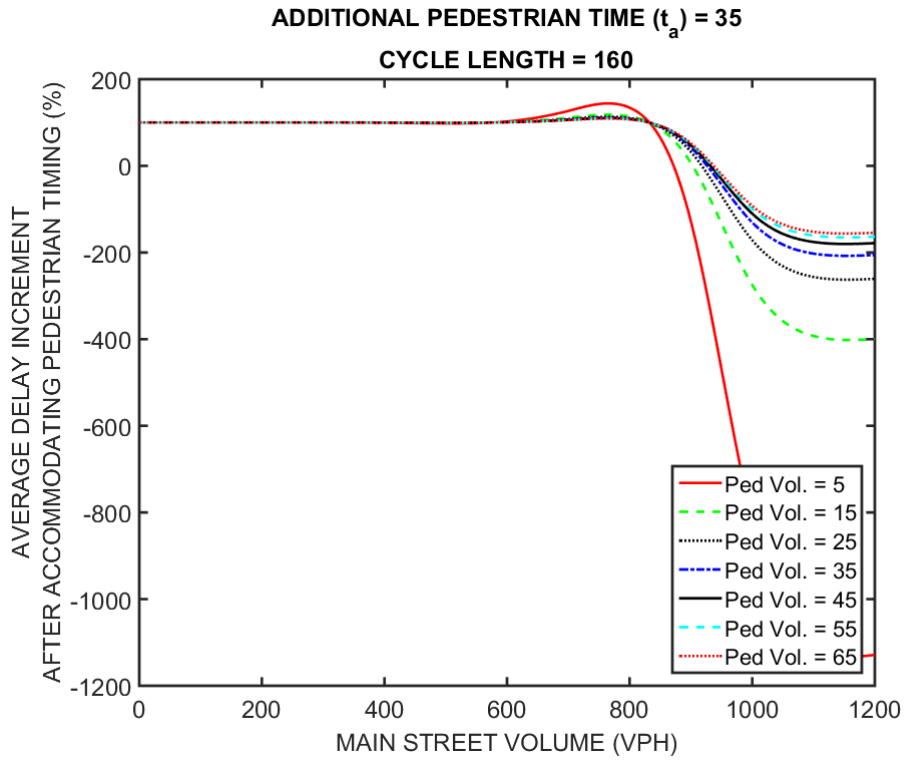

Figure 11: t_a analysis over cycle length, pedestrian and vehicle volume, $t_a = 35$ sec

Using similar figures to Figures 11 to 13, the vehicle and pedestrian volume thresholds for accommodating pedestrian time can be extracted and summarized into reference diagrams similar to the one shown in Figure 14 for any condition. The left side of each curve at this figure shows the non-accommodating area and the right side shows the accommodating area. For example, in Figure 14, if pedestrian volume is five persons per hour and t_a is 55 seconds, accommodation of pedestrian timing into signal coordination is more beneficial for volume more than 400 vph.

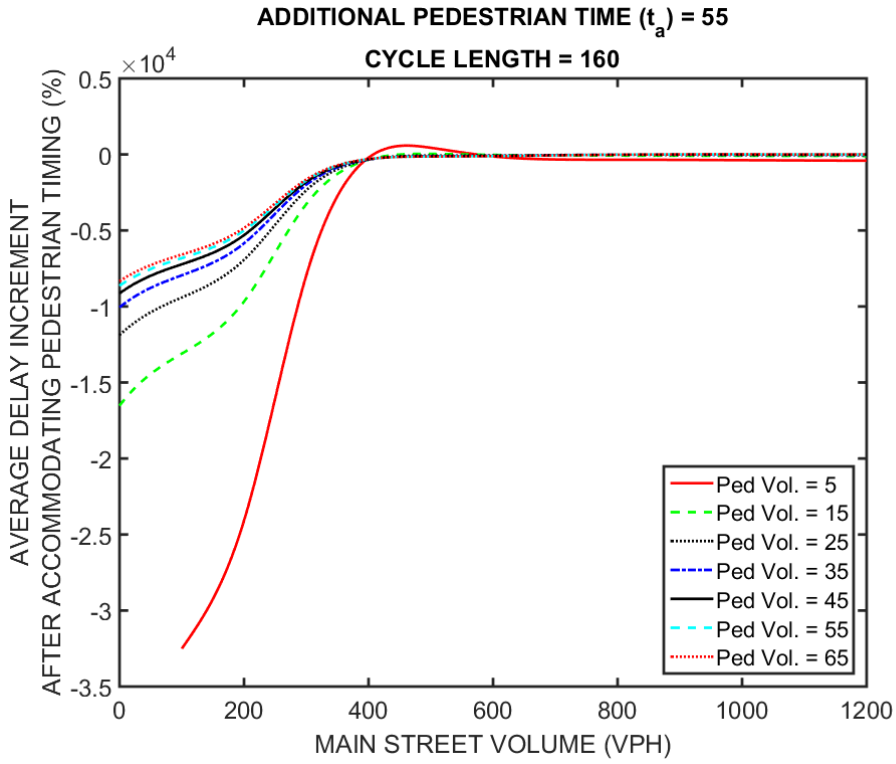

Figure 12: t_a analysis over cycle length, pedestrian and vehicle volume, $t_a = 45$ sec

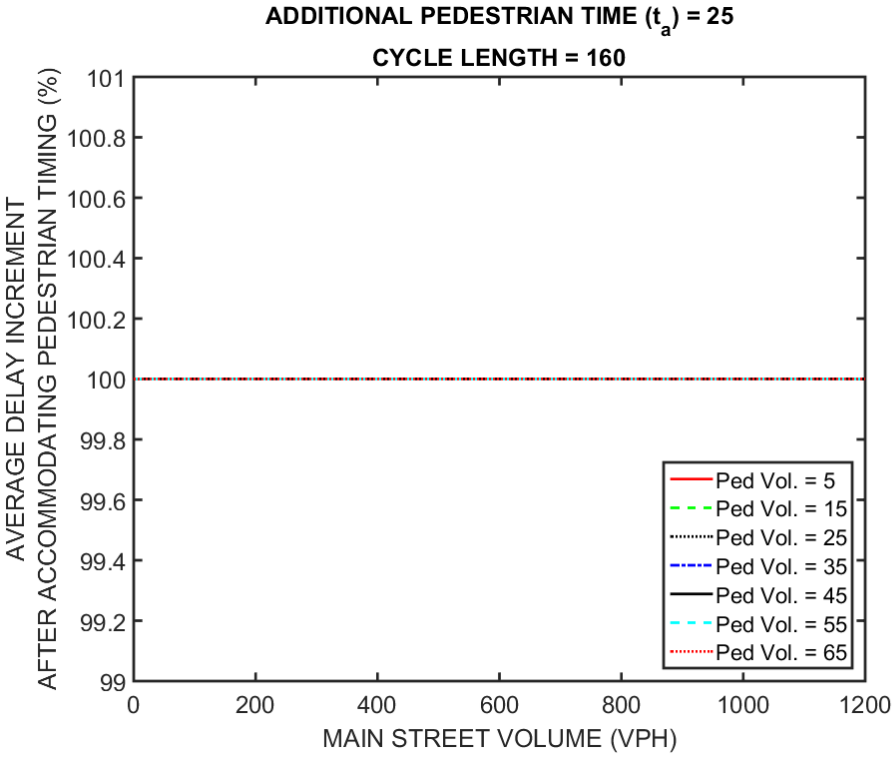

Figure 13: t_a analysis over cycle length, pedestrian and vehicle volume, $t_a = 25$ sec

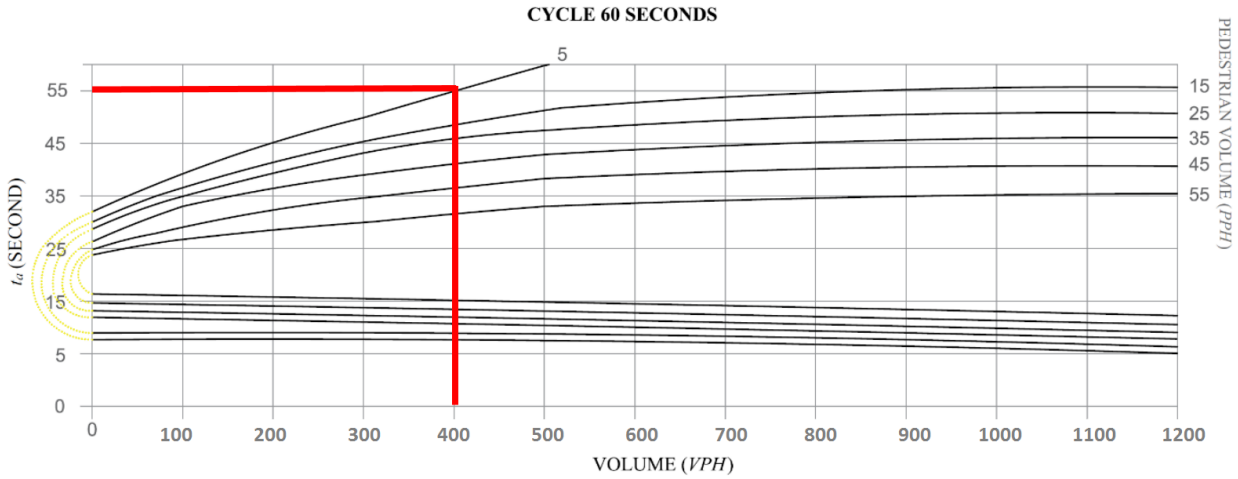

Figure 14: Volume threshold for accommodating pedestrian timing into signal coordination, cycle length 60 seconds. Left side of each curve shows not accommodation area and right side accommodation area.

SUMMARY

Accommodating pedestrian timing in coordinated signal systems implies that the cycle length must be long enough to accommodate the Walk and Flash-Don't-Walk (called pedestrian clearance interval) for the side street. The advantage of this treatment is that a signal will always remain in coordination regardless of whether there is a pedestrian crossing or not. The major disadvantage is the resulting longer cycle length, which may be operationally inefficient when vehicular traffic volume is low. Unnecessary long vehicle delays can result. The other alternative is to not accommodate pedestrian timing, i.e., the side street phase split is shorter than what is needed for pedestrian crossing. Although this alternative has no pedestrian safety degradation (i.e., pedestrians still receive the same WALK and Flash-Don't-Walk times during a crossing), the signal may go into transition, which will disrupt coordination, negatively affecting the system efficiency. In order to determine what pedestrian timing alternative is the best, a mathematical model was provided that relates vehicle volume, pedestrian volume, and network conditions to the delay caused by adding pedestrian timing.

To facilitate using the mathematical model, a software has been developed based on it called PeTASC (Pedestrian Timing Accommodation into Signal Coordination) that can be accessed by scanning the QR code or following the link in Figure 7.

The purpose of this software is to provide practitioners an easy way to determine when accommodation (A) of pedestrian timing into coordination is preferable over non-accommodating (NA). With this software, practitioners input cycle length (C), volume (v), required pedestrian timing (RPT), and other signal parameters. The software will then show them which plan, A or NA, has lower delay.

To validate the mathematical model and its software, a code was written in COM interface of VISSIM (appendix A) to model a wide range of vehicle and pedestrian volume combinations. These combinations formed 3,456 scenarios. For each scenario, two methods of accommodation and non-accommodation of pedestrian timing were performed. The outputs of these scenarios were compared with the ones obtained from the mathematical model.

The mathematical model does not consider the distance between intersections as a significant variable. To check if the distance of intersections can affect the results, all 3,456 scenarios were retested by multiplying the distance of intersections by factors 0.5 and 1.5. The results of these three sets of scenarios were compared. Analysis on results shows that the distance of intersections does not affect the results significantly.

REFERENCES

Federal Highway Administration (2008). Traffic Signal Timing Manual, Report FHWA-HOP-08-024.

Parsonson, P. (1992). Signal Timing Improvement Practices, NCHRP 172, Transportation Research Board.

Petzold, R. (1977). Urban Intersection Improvements for Pedestrian Safety--Volume III: Signal Timing for the Pedestrian, FHWA Report RD-77.

Tian, Z. Z., T. Urbanik, K. K. Kacir, M. A. Vandehey, and H. Long (2000). Pedestrian Timing Treatment for Coordinated Signal Systems. Proc., 2nd International Conference on Transportation and Traffic Studies, Beijing, China, ASCE, 2000, pp. 533–540.

Tian, Z. and F. Xu (2006). Modeling the Effects of Pedestrians on Intersection Capacity and Delay with Actuated Signal Control, Proceedings of the 5th International Symposium on Highway Capacity, Yokohama, Japan, July 2006.

Tian, Z., K. Kacir, M. Vandehay, and H. Long (1999). Signal Timing Strategies in Dealing with Pedestrian Crossings. In Proc., 69th ITE Annual Meeting, Las Vegas, Nev., Aug. 1999.

Tian, Z., Urbanik, T., Engelbrecht, R., and Balke, K. (2001). Pedestrian Timing Alternatives and Impacts on Coordinated Signal Systems under Split-Phasing Operations, Transportation Research Record 1748, pp. 46-54.

Tian, Z. (2004). Pedestrian Timing Treatment for Coordinated Signal Systems, Proceedings of International Conference on Traffic and Transportation Studies.

Zegeer, C., Cynecki, M., and Opiela, K. (1982). Effect of Pedestrian Signals and Signal Timing on Pedestrian Accidents, Transportation Research Record 847, pp. 62-72.

Zegeer, C., Cynecki, M., and Opiela, K. (1984). Evaluation of Innovative Pedestrian Signalization Alternatives, Transportation Research Record 959, pp. 7-18.

Appendix A: COM interface C# code

Lib File:

```
using VISSIMLIB;
using VISSIMCOMLIB;
```

MainFunc:

```
    public partial class MainWindow : Window
    {
        string InpxFilename;
        string LayxFilename;
        Vissim _m_Vissim;
        int _m_simperiod = 3600;
        int _m_Ped_num=6; //6
        int _m_MainSt_num=10;//10

        public MainWindow()
        {
            InitializeComponent();

        }

        private void _open_file_Click(object sender, RoutedEventArgs e)
        {
            Microsoft.Win32.OpenFileDialog dlg = new Microsoft.Win32.OpenFileDialog();

            // Set filter for file extension and default file extension
            dlg.Filter = "VISSIM Files (*.inpx; *.layx; *.txt)|*.inpx;*.layx;*.txt";
            dlg.Multiselect = false;

            // Display OpenFileDialog by calling ShowDialog method
            Nullable<bool> result = dlg.ShowDialog();

            // Get the selected file name and display in a TextBox
            if (result == true)
            {
                foreach (String file in dlg.FileNames)
                {
                    // Create a PictureBox.
                    if (file.Contains(".inpx"))
                    {
                        InpxFilename = file;
                        LayxFilename = file.Replace(".inpx", ".layx");
                    }

                }

                if (InpxFilename == null)
                {
                    MessageBox.Show("Do not add the *.inpx file sucessfully.",
"Important Note");
                    return;
                }


                _m_Vissim = new Vissim();
```

```
                // If you have installed muliple Vissim Versions, you have to set the
reference to the Vissim Version you want to open.
                VISSIMCOMFunction.LoadNet(_m_Vissim, InpxFilename);
                VISSIMCOMFunction.LoadLayout(_m_Vissim, LayxFilename);

                _m_simperiod = VISSIMCOMFunction.GetSimPeriod( _m_Vissim);

                MessageBox.Show("Completed!");

            }

        }


        public void SIMRun(int i, int j,int m)
        {
            VISSIMCOMFunction.SetMaxSimSpeed(_m_Vissim);
            VISSIMCOMFunction.RunContinuous(_m_Vissim);
            VISSIMCOMFunction.StopSim(_m_Vissim);
        }

        private void _b_start_Click(object sender, RoutedEventArgs e)
        {
            int count = 0;
            int startvolume_MajorSt = 200;
            if (_UI_method_2.IsChecked == true)
                startvolume_MajorSt = 100;

            for (int i = 0; i < _m_Ped_num; i++)
            {
                for (int j = 0; j < _m_MainSt_num; j++)
                {

                    int PedNB = (i + 1) * 5;
                    int MajorSt = startvolume_MajorSt + (j * 200);
                    int OtherSt = MajorSt * 30 / 100;

                    oSheet.Cells[count + 2, 20] = PedNB;
                    oSheet.Cells[count + 2, 21] = MajorSt;
                    oSheet.Cells[count + 2, 22] = OtherSt;

                    count++;

                    do
                    {
                        VISSIMCOMFunction.SetVolume(_m_Vissim, 2, OtherSt);
                    }
                    while ( VISSIMCOMFunction.GetVolume(_m_Vissim,2) != OtherSt);

                    do
                    {
                        VISSIMCOMFunction.SetVolume(_m_Vissim, 8, PedNB);
                    }
                    while ( VISSIMCOMFunction.GetVolume(_m_Vissim,8) != PedNB);

                    do
                    {
                        VISSIMCOMFunction.SetVolume(_m_Vissim, 10, OtherSt);
                    }
                    while ( VISSIMCOMFunction.GetVolume(_m_Vissim,10) != OtherSt);

                    do
```

```
                    {
                        VISSIMCOMFunction.SetVolume(_m_Vissim, 11, MajorSt);
                    }
                    while ( VISSIMCOMFunction.GetVolume(_m_Vissim,11) != MajorSt);

                    do
                    {
                        VISSIMCOMFunction.SetVolume(_m_Vissim, 12, OtherSt);
                    }
                    while ( VISSIMCOMFunction.GetVolume(_m_Vissim,12) != OtherSt);

                    SIMRun(PedNB, MajorSt, OtherSt);
                }
            }
            MessageBox.Show("Completed!");
        }

        private void _Exit_Click(object sender, RoutedEventArgs e)
        {
            if (_m_Vissim != null)
                VISSIMCOMFunction.Exit(_m_Vissim);
            this.Close();

        }


    }
}
```

Appendix B: PeTASC MATLAB code

```
% This function calculates gbji and grji: The time interval at the
beginning
% of green time of main street at Intersection i that vehicles dispatch as
% bunched/random from the first intersection in Cycle i during transition
(sec).
% By Ali Gholami------12/29/2015

function [gb1s,gr1s, gb2s, gr2s, gb1l, gr1l, gb2l, gr2l] = gbrji(i, c, g,
ta, vm, fs, mu)


r = c - g;
betas = ceil(ta/c/mu);
betal = ceil((c-ta)/c/mu);
alphas = -ta / betas;
alphal = (c-ta) / betal;
mus = 1 - ((c + alphas) / c);%the real percentage of cycle length that is
subtracted
mul = ((c + alphal) / c) - 1;%the real percentage of cycle length that is
added
gs = g - mus * g;
gl = g + mul * g;

[L1s,L2s, L1l, L2l] = Lji(i, c, g, ta, mu);
[U1s,U2s, U1l, U2l] = Uji(i, c, g, ta, mu);

gd1s = L1s - U1s;
gd2s = L2s - U2s;
gd1l = L1l - U1l;
gd2l = L2l - U2l;

if (r * vm / fs) < g
    gb = r * vm / fs;
else
    gb = g;
end

gr = g - gb;

%Shortening
if U1s == 0 && gb < gd1s
    gb1s = gb;
elseif U1s == 0 && gb >= gd1s
    gb1s = gd1s;
elseif U1s > 0 && gd1s == g
    gb1s = gb;
elseif U1s > 0 && gd1s <= g && gr >= gd1s
    gb1s = 0;
else
    gb1s = gd1s - gr;
end

if U1s == 0 && gb < gd1s
    gr1s = gd1s - gb;
elseif U1s == 0 && gb >= gd1s
    gr1s = 0;
```

```
elseif U1s > 0 && gd1s == g
    gr1s = gr;
elseif U1s > 0 && gd1s <= g && gr >= gd1s
    gr1s = gd1s;
else
    gr1s = gr;
end

if U2s == 0 && gb < gd2s
    gb2s = gb;
elseif U2s == 0 && gb >= gd2s
    gb2s = gd2s;
elseif U2s > 0 && gd2s == gs
    gb2s = gb;
elseif U2s > 0 && gd2s <= gs && gr >= gd2s
    gb2s = 0;
else
    gb2s = gd2s - gr;
end

if U2s == 0 && gb < gd2s
    gr2s = gd2s - gb;
elseif U2s == 0 && gb >= gd2s
    gr2s = 0;
elseif U2s > 0 && gd2s == gs
    gr2s = gr;
elseif U2s > 0 && gd2s <= gs && gr >= gd2s
    gr2s = gd2s;
else
    gr2s = gr;
end


% lengthening
if U1l == 0 && gb < gd1l
    gb1l = gb;
elseif U1l == 0 && gb >= gd1l
    gb1l = gd1l;
elseif U1l > 0 && gd1l == g
    gb1l = gb;
elseif U1l > 0 && gd1l <= g && gr >= gd1l
    gb1l = 0;
else
    gb1l = gd1l - gr;
end

if U1l == 0 && gb < gd1l
    gr1l = gd1l - gb;
elseif U1l == 0 && gb >= gd1l
    gr1l = 0;
elseif U1l > 0 && gd1l == g
    gr1l = gr;
elseif U1l > 0 && gd1l <= g && gr >= gd1l
    gr1l = gd1l;
else
    gr1l = gr;
end

if U2l == 0 && gb < gd2l
    gb2l = gb;
```

```
elseif U2l == 0 && gb >= gd2l
    gb2l = gd2l;
elseif U2l > 0 && gd2l == gl
    gb2l = gb;
elseif U2l > 0 && gd2l <= gl && gr >= gd2l
    gb2l = 0;
else
    gb2l = gd2l - gr;
end

if U2l == 0 && gb < gd2l
    gr2l = gd2l - gb;
elseif U2l == 0 && gb >= gd2l
    gr2l = 0;
elseif U2l > 0 && gd2l == gl
    gr2l = gr;
elseif U2l > 0 && gd2l <= gl && gr >= gd2l
    gr2l = gd2l;
else
    gr2l = gr;
end
```

```
% This function calculates the number of cycles that go into transition
before another pedestrian call occurs.
% By Ali Gholami------12/29/2015
function [gammas, gammal] = gamma1(c, ta, vp, mu)


betas = ceil(ta/c/mu);
betal = ceil((c-ta)/c/mu);



P1 = 1- exp(-(vp/3600)*c); %Probability of having at least one pedestrian
call during one cycle

if (1 / P1) <= betas
    gammas = ceil(1 / P1); %The number of cycles that go into transition
before another pedestrian call occurs: Shortening method
else
    gammas = betas;
end

if (1 / P1) <= betal
    gammal = ceil(1 / P1); %The number of cycles that go into transition
before another pedestrian call occurs: Lengthening method
else
    gammal = betal;
end
```

```
% This function calculates hourly delay caused by transition for
% both shortening and lengthening transition methods at Intersection 1
% (intersection with pedestrian call) and its next intersection
% By Ali Gholami------12/29/2015


function [dns_1, dns_2, dns_total, dnl_1, dnl_2, dnl_total] = dn(c, g, gl,
ta, vm, fs, vp, vs, vl, mu, rho, E)


ta1 = ta; % this ta will be used for side street
% Modified t_a for non-accommodating PT at semi-actuated coordination
vlc = vl * c; % Main street left turn volume per cycle
clc = fs * gl; % Capacity of left turn (vehicle per cycle)
if (vlc / clc) >= 1 || ((vlc / clc) * gl + E) >= gl
    ta = ta;
elseif vl == 0
    ta = ta - (min (gl, ta));
elseif ((vlc / clc) < 1 && (vlc / clc) > 0) && ((vlc / clc) * gl + E) < gl
    ta = ta - (min ((gl - ((vlc / clc) * gl) + E), ta));
end



[gammas, gammal] = gamma1(c, ta, vp, mu);

% Calculating delay of one transition period due to non-accommodating PT at
the
% Intersection 2 with shortening and Lengthening transition methods
Zn1cs_1 = 0;
Zn1cs_2 = 0;
Zn1cl_1 = 0;
Zn1cl_2 = 0;

%Shortening
if gammas == 0
    Zn1cs_1 = - (vs*rho*(ta1^2)/2);
    Zn1cs_2 = 0;
else
    for i = 1:gammas
        [L1s,L2s, L1l, L2l] = Lji(i, c, g, ta, mu);
        [U1s,U2s, U1l, U2l] = Uji(i, c, g, ta, mu);
        [gb1s,gr1s, gb2s, gr2s, gb1l, gr1l, gb2l, gr2l] = gbrji(i, c, g,
ta, vm, fs, mu);
        Zn1cs_1 = Zn1cs_1 + (gb1s*fs*(L1s+U1s+gr1s)/2) +...
            (gr1s*vm*(L1s+U1s-gb1s)/2); %Delay of one
        %transition period due to non-accommodating PT at the Intersection
1 with shortening transition method
        Zn1cs_2 = Zn1cs_2 + (gb2s*fs*(L2s+U2s+gr2s)/2) +...
            (gr2s*vm*(L2s+U2s-gb2s)/2);
    end
    Zn1cs_1 = Zn1cs_1 - (vs*rho*(ta^2)/2);
end

%Lengthening
if gammal == 0
    Zn1cl_1 = - (vs*rho*(ta1^2)/2);
    Zn1cl_2 = 0;
else
```

```
    for i = 1:gammal
        [L1s,L2s, L1l, L2l] = Lji(i, c, g, ta, mu);
        [U1s,U2s, U1l, U2l] = Uji(i, c, g, ta, mu);
        [gb1s,gr1s, gb2s, gr2s, gb1l, gr1l, gb2l, gr2l] = gbrji(i, c, g,
ta, vm, fs, mu);
        Zn1cl_1 = Zn1cl_1 + (gb1l*fs*(L1l+U1l+gr1l)/2) +...
            (gr1l*vm*(L1l+U1l-gb1l)/2); %Delay of one
        %transition period due to non-accommodating PT at the Intersection
1 with lengthening transition method
        Zn1cl_2 = Zn1cl_2 + (gb2l*fs*(L2l+U2l+gr2l)/2) +...
            (gr2l*vm*(L2l+U2l-gb2l)/2);
    end
    Zn1cl_1 = Zn1cl_1 - (vs*rho*(ta1^2)/2);
end


[nts, ntl] = nt(c, ta, vp, mu);%Number of transition periods per hour for
both shortening and lengthening transition methods
if gammas == 0
    dns_1 = Zn1cs_1;
    dns_2 = 0;
    dns_total = 0;
else
    dns_1 = nts * Zn1cs_1; %hourly delay caused by transition for
shortening transition methods at the Intersection 1
    dns_2 = nts * Zn1cs_2; %hourly delay caused by transition for
shortening transition methods at the Intersection 2
    dns_total = dns_1 + dns_2; %Total Shortening
end

if gammal == 0
    dnl_1 = Zn1cl_1;
    dnl_2 = 0;
    dnl_total = 0;
else
    dnl_1 = ntl * Zn1cl_1; %hourly delay caused by transition for
lengthening transition methods at the Intersection 1
    dnl_2 = ntl * Zn1cl_2; %hourly delay caused by transition for
lengthening transition methods at the Intersection 2
    dnl_total = dnl_1 + dnl_2; %Total Lengthening
end
```

```
% This function calculates hourly delay caused by accommodating pedestrian
% timing
% By Ali Gholami------12/30/2015


function [da1, dan, da_total] = da(c, n, g, ta, vm, fs, vp, vs, rho)

P1 = 1- exp(-(vp/3600)*c); %Probability of having at least one pedestrian
call during one cycle
ga = g - ta;
greq = vm * c / fs;
ta = ta * P1;

% First intersection
da1 = ta^2 * (vm - (rho * vs)) * 1800 / c; %Delay caused by adding t_a to
side street green time (veh-sec) at Intersection 1.


%Next intersections
dan = 0;
for i = 2:n
    if greq <= ga
        dan = dan + (ta^2 * (0 - (rho * vs)) * 1800 / c);
    else
        dan = dan + ((min (ta, (greq - ga)))^2 * (fs - (rho * vs)) * 1800 /
c);
    end
end

da_total = da1 + dan;
```

```
% This function calculates Average cycle length during one hour (sec) for
% both shortening and lengthening transition methods
% By Ali Gholami------12/29/2015

function [cas, cal] = ca(c, ta, vp, mu)
% clear
% c = 70;
% ta = 40;
% vp = 5;
% mu = .1;

betas = ceil(ta/c/mu);
betal = ceil((c-ta)/c/mu);
cs = c - ta/betas;
cl = c + ta/betal;
P1 = 1- exp(-(vp/3600)*c); %Probability of having at least one pedestrian
call during one cycle
[gammas, gammal] = gamma1(c, ta, vp, mu);
%Average cycle length during one hour with shortening transition method
(sec)
if 1/P1 <= betas
    cas = ((c+ta)+((gammas-1)*(cs)))/gammas;
else
    cas = (((c+ta)+((betas-1)*(cs)))/betas)*P1*betas + c*(1-(P1*betas));
end
%Average cycle length during one hour with lengthening transition method
(sec)
if 1/P1 <= betal
    cal = ((c+ta)+((gammal-1)*(cl)))/gammal;
else
    cal = (((c+ta)+((betal-1)*(cl)))/betal)*P1*betal + c*(1-(P1*betal));
end
```

```
% This function calculates Number of transition periods per hour for
% both shortening and lengthening transition methods
% By Ali Gholami------12/29/2015

function [nts, ntl] = nt(c, ta, vp, mu)

betas = ceil(ta/c/mu);
betal = ceil((c-ta)/c/mu);
P1 = 1- exp(-(vp/3600)*c); %Probability of having at least one pedestrian
call during one cycle
[gammas, gammal] = gamma1(c, ta, vp, mu);
%Average cycle length during one hour with (sec)
[cas, cal] = ca(c, ta, vp, mu);

if 1/P1 <= betas
    nts = 3600 / cas / gammas;
else
    nts = 3600 / cas * P1;
end
if 1/P1 <= betal
    ntl = 3600 / cal / gammal;
else
    ntl = 3600 / cal * P1;
end
```

```
% This function calculates Lj^i: The time distance of the lower bound of
% delayed green to beginning of the next green time at Intersection j in
Cycle i during transition
% By Ali Gholami------12/29/2015

function [L1s,L2s, L1l, L2l] = Lji(i, c, g, ta, mu)

r = c - g;
betas = ceil(ta/c/mu);
betal = ceil((c-ta)/c/mu);
alphas = (-ta) / betas;
alphal = (c-ta) / betal;
mus = 1 - ((c + alphas) / c);%the real percentage of cycle length that is
subtracted
mul = ((c + alphal) / c) - 1;%the real percentage of cycle length that is
added
rs = r - mus * r;
rl = r + mul * r;


if (ta + (i - 1) * alphas) <= rs
    L1s = ta + (i - 1) * alphas;
else
    L1s = rs;
end

if (ta + (i - 1) * alphal) <= rl
    L1l = ta + (i - 1) * alphal;
else
    L1l = rl;
end

if (c - ta - (i - 1) * alphas) <= r
    L2s = c - ta - (i - 1) * alphas;
else
    L2s = r;
end

if (c - ta - (i - 1) * alphal) <= r
    L2l = c - ta - (i - 1) * alphal;
else
    L2l = r;
end
```

```
% This function calculates Uj^i: The time distance of the upper bound of
% delayed green to beginning of the next green time at Intersection j in
Cycle i during transition
% By Ali Gholami------12/29/2015

function [U1s,U2s, U1l, U2l] = Uji(i, c, g, ta, mu)

betas = ceil(ta/c/mu);
betal = ceil((c-ta)/c/mu);
alphas = (-ta) / betas;
alphal = (c-ta) / betal;
mus = 1 - ((c + alphas) / c);%the real percentage of cycle length that is
subtracted
mul = ((c + alphal) / c) - 1;%the real percentage of cycle length that is
added
gs = g - mus * g;
gl = g + mul * g;


if (ta + (i - 1) * alphas) > g
    U1s = ta + (i - 1) * alphas - g;
else
    U1s = 0;
end

if (ta + (i - 1) * alphal) > g
    U1l = ta + (i - 1) * alphal - g;
else
    U1l = 0;
end

if (c - ta - (i - 1) * alphas -g) > gs
    U2s = c - ta - (i - 1) * alphas -g;
else
    U2s = 0;
end

if (c - ta - (i - 1) * alphal -g) > gl
    U2l = c - ta - (i - 1) * alphal -g;
else
    U2l = 0;
end
```

```
% This function compares hourly delay caused by accommodating pedestrian
% timing and non-accommodating PT
% By Ali Gholami------12/30/2015

function [y1, y2, y3, comparison, which_method] = compare2 (c, gpercent,
glpercent, ta, vp, vlpercent, vspercent, E, n, fs, vmto)

g = gpercent * c;
gl = glpercent * c; % Left turn green time interval (sec)
vm = 400 / 3600;
vs = vmto * vspercent ;
fs = fs / 3600;
vl = vmto * vlpercent;

comparison = [];
which_method = [];

for tt = 0.05:0.05:.3
    mu = tt;
    y1 = [];
    y2 = [];
    y3 = [];
    for ii = 0:.25:1
        rho = ii;

        [dns_1, dns_2, dns_total, dnl_1, dnl_2, dnl_total] = dn(c, g, gl,
ta, vm, fs, vp, vs, vl, mu, rho, E);
        [da1, dan, da_total] = da(c, n, g, ta, vm, fs, vp, vs, rho);
        besttransition = min (dns_total, dnl_total);
        accomm_delay_increase = (da_total - besttransition)/da_total*100;
        a = [ta; c; vm*3600; vs*3600; vl*3600; vp; mu; rho; dns_total;
dnl_total; da_total];
        b = [ta; c; vm*3600; vs*3600; vl*3600; vp; mu; rho;
accomm_delay_increase];
        comparison = [comparison, a]; % Comparison of shortening,
lengthening and accommodating
        which_method = [which_method, b];
        y1 = [y1, dns_total];
        y2 = [y2, dnl_total];
        y3 = [y3, da_total];
    end
    x=0:.25:1;
    figure
    plot(x, y1,x,y2,'-.k',x,y3,'--r')
    legend('Shortening','Lengthening','Accommodating PT')
    xlabel('\rho (weight of side street volume)')
    ylabel('Delay caused by adding PT (veh-sec)')
    title ({['Maximum Percentage Permitted to']; ['Add to/Subtract from
Cycle Length, \mu = ', num2str(mu*100), '%']})
end
```

```
% This function compares hourly delay caused by accommodating pedestrian
% timing and non-accommodating PT
% By Ali Gholami------12/30/2015

function [y1, y2, y3, comparison, which_method] = compare3 (c, gpercent,
glpercent, ta, vp, vlpercent, vspercent, mu, rho, E, n, fs, vmfrom,
vminterval, vmto)


fs = fs / 3600;
comparison = [];
which_method = [];
y1 = [];
y2 = [];
y3 = [];
for ii = vmfrom:vminterval:vmto % Main treet volume
    vm = ii / 3600;
    vs = ii * vspercent / 3600;
    vl = ii * vlpercent / 3600;

    g = gpercent * c;
    gl = glpercent * c;

    [dns_1, dns_2, dns_total, dnl_1, dnl_2, dnl_total] = dn(c, g, gl, ta,
vm, fs, vp, vs, vl, mu, rho, E);
    [da1, dan, da_total] = da(c, n, g, ta, vm, fs, vp, vs, rho);
    besttransition = min (dns_total, dnl_total);
    accomm_delay_increase = (da_total - besttransition)/da_total*100;
    a = [ta; c; vm*3600; vs*3600; vl*3600; vp; mu; rho; dns_total;
dnl_total; da_total];
    b = [ta; c; vm*3600; vs*3600; vl*3600; vp; mu; rho;
accomm_delay_increase];
    comparison = [comparison, a]; % Comparison of shortening, lengthening
and accommodating
    which_method = [which_method, b];

    y1 = [y1, dns_total];
    y2 = [y2, dnl_total];
    y3 = [y3, da_total];

end
```

```
% This function compares hourly delay caused by accommodating pedestrian
% timing and non-accommodating PT
% By Ali Gholami------12/30/2015

function [y1, comparison, which_method] = compare4 (gpercent, glpercent,
vlpercent, vspercent, mu, rho, E, n, fs, vmfrom, vminterval, vmto)

comparison = [];
which_method = [];
fs = fs / 3600;

for kk = 60:20:200
    c = kk;

    for tt = 5:10:55
        ta = tt;
        y1 = [];
        for jj = 5:10:65 % Pedestrian
            vp = jj;
            best_tran_matrix = [];
            for ii = vmfrom:vminterval:vmto % Main treet volume
                vm = ii / 3600;
                vs = ii * vspercent / 3600;

                vl = ii * vlpercent / 3600;
                g = gpercent * c;
                gl = glpercent * c;

                [dns_1, dns_2, dns_total, dnl_1, dnl_2, dnl_total] = dn(c,
g, gl, ta, vm, fs, vp, vs, vl, mu, rho, E);
                [da1, dan, da_total] = da(c, n, g, ta, vm, fs, vp, vs,
rho);
                besttransition = min (dns_total, dnl_total);
                accomm_delay_increase = (da_total -
besttransition)/da_total*100;
                best_tran_matrix = [best_tran_matrix,
accomm_delay_increase];
                a = [ta; c; vm*3600; vs*3600; vl*3600; vp; mu; rho;
dns_total; dnl_total; da_total];
                b = [ta; c; vm*3600; vs*3600; vl*3600; vp; mu; rho;
accomm_delay_increase];
                comparison = [comparison, a]; % Comparison of shortening,
lengthening and accommodating
                which_method = [which_method, b];

            end
            y1 = [y1; best_tran_matrix];

        end
        x=vmfrom:vminterval:vmto;
        curve1 = fit(x',(y1(1,:))','smoothingspline');
        curve2 = fit(x',(y1(2,:))','smoothingspline');
        curve3 = fit(x',(y1(3,:))','smoothingspline');
        curve4 = fit(x',(y1(4,:))','smoothingspline');
        curve5 = fit(x',(y1(5,:))','smoothingspline');
        curve6 = fit(x',(y1(6,:))','smoothingspline');
        curve7 = fit(x',(y1(7,:))','smoothingspline');
        figure
        plot(curve1)
        hold on
```

```
        plot(curve2, '--g')
        plot(curve3, ':k')
        plot(curve4, '-.b')
        plot(curve5, 'k')
        plot(curve6, '--c')
        plot(curve7, ':r')
        %plot(x, y1(1,:),x, y1(2,:),x, y1(3,:),x, y1(4,:),x, y1(5,:),x,
y1(6,:),x, y1(7,:))
        legend('Ped Vol. = 5','Ped Vol. = 15','Ped Vol. = 25', 'Ped Vol. =
35','Ped Vol. = 45','Ped Vol. = 55','Ped Vol. = 65','Location','southeast')
        xlabel({'MAIN STREET VOLUME (VPH)'})
        ylabel({'AVERAGE DELAY INCREMENT'; 'AFTER ACCOMMODATING PEDESTRIAN
TIMING (%)'})
        title ({['ADDITIONAL PEDESTRIAN TIME (t_{a}) = ', num2str(ta)];
['CYCLE LENGTH = ', num2str(c)]})

    end
end
```

```
function varargout = PeTASC(varargin)
% PETASC MATLAB code for PeTASC.fig
%      PETASC, by itself, creates a new PETASC or raises the existing
%      singleton*.
%
%      H = PETASC returns the handle to a new PETASC or the handle to
%      the existing singleton*.
%
%      PETASC('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in PETASC.M with the given input arguments.
%
%      PETASC('Property','Value',...) creates a new PETASC or raises the
%      existing singleton*.  Starting from the left, property value pairs
are
%      applied to the GUI before PeTASC_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to PeTASC_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help PeTASC

% Last Modified by GUIDE v2.5 06-Jan-2016 15:58:31

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @PeTASC_OpeningFcn, ...
                   'gui_OutputFcn',  @PeTASC_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before PeTASC is made visible.
function PeTASC_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to PeTASC (see VARARGIN)

% Choose default command line output for PeTASC
handles.output = hObject;

% Update handles structure
```

```
guidata(hObject, handles);

% UIWAIT makes PeTASC wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = PeTASC_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
c = str2num(get(handles.edit1,'String'));
gpercent = str2num(get(handles.edit2,'String'));
glpercent = str2num(get(handles.edit3,'String'));
ta = str2num(get(handles.edit4,'String'));
vp = str2num(get(handles.edit5,'String'));
vlpercent = str2num(get(handles.edit6,'String'));
vspercent = str2num(get(handles.edit7,'String'));
mu = str2num(get(handles.edit8,'String'));
rho = str2num(get(handles.edit9,'String'));
E = str2num(get(handles.edit10,'String'));
n = str2num(get(handles.edit11,'String'));
fs = str2num(get(handles.edit12,'String'));
vmfrom = str2num(get(handles.edit13,'String'));
vminterval = str2num(get(handles.edit14,'String'));
vmto = str2num(get(handles.edit15,'String'));

[y1, y2, y3, comparison, which_method] = compare3 (c, gpercent, glpercent,
ta, vp, vlpercent, vspercent, mu, rho, E, n, fs, vmfrom, vminterval, vmto)
x=vmfrom:vminterval:vmto;
% figure
% plot(x, y1,x,y2,x,y3)
% legend('Shortening','Lengthening','Accommodating PT')
% xlabel('Volume (vehicle per approach')
% ylabel('Delay caused by adding PT (veh-sec)')
% y=myfunction(x);
plot(handles.axes1,x, y1,x,y2,'-.k',x,y3, '--r');
legend('Shortening','Lengthening','Accommodating PT')
xlabel('Volume (vehicle per approach)')
ylabel('Delay caused by adding PT (veh-sec)')
title ({'COMPARISON OF PEDESTRIAN TIMING'; 'ACCOMMODATION METHODS'})
% figure(1);
% plot(x,y);



function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a
double


% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a
double


% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3 as a
double
```

```
% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit4_Callback(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit4 as text
%        str2double(get(hObject,'String')) returns contents of edit4 as a
double


% --- Executes during object creation, after setting all properties.
function edit4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit5_Callback(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit5 as text
%        str2double(get(hObject,'String')) returns contents of edit5 as a
double


% --- Executes during object creation, after setting all properties.
function edit5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
```

```
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit6_Callback(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit6 as text
%        str2double(get(hObject,'String')) returns contents of edit6 as a
double


% --- Executes during object creation, after setting all properties.
function edit6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit7_Callback(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit7 as text
%        str2double(get(hObject,'String')) returns contents of edit7 as a
double


% --- Executes during object creation, after setting all properties.
function edit7_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit8_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit8 as text
%        str2double(get(hObject,'String')) returns contents of edit8 as a
double


% --- Executes during object creation, after setting all properties.
function edit8_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit9_Callback(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit9 as text
%        str2double(get(hObject,'String')) returns contents of edit9 as a
double


% --- Executes during object creation, after setting all properties.
function edit9_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit10_Callback(hObject, eventdata, handles)
% hObject    handle to edit10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit10 as text
%        str2double(get(hObject,'String')) returns contents of edit10 as a
double
```

```
% --- Executes during object creation, after setting all properties.
function edit10_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit11_Callback(hObject, eventdata, handles)
% hObject    handle to edit11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit11 as text
%        str2double(get(hObject,'String')) returns contents of edit11 as a
double


% --- Executes during object creation, after setting all properties.
function edit11_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit12_Callback(hObject, eventdata, handles)
% hObject    handle to edit12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit12 as text
%        str2double(get(hObject,'String')) returns contents of edit12 as a
double


% --- Executes during object creation, after setting all properties.
function edit12_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
```

```
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit13_Callback(hObject, eventdata, handles)
% hObject    handle to edit15 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit15 as text
%        str2double(get(hObject,'String')) returns contents of edit15 as a
double


% --- Executes during object creation, after setting all properties.
function edit13_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit15 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit14_Callback(hObject, eventdata, handles)
% hObject    handle to edit14 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit14 as text
%        str2double(get(hObject,'String')) returns contents of edit14 as a
double


% --- Executes during object creation, after setting all properties.
function edit14_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit14 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function edit15_Callback(hObject, eventdata, handles)
% hObject    handle to edit15 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit15 as text
%        str2double(get(hObject,'String')) returns contents of edit15 as a
double


% --- Executes during object creation, after setting all properties.
function edit15_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit15 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit16_Callback(hObject, eventdata, handles)
% hObject    handle to edit14 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit14 as text
%        str2double(get(hObject,'String')) returns contents of edit14 as a
double


% --- Executes during object creation, after setting all properties.
function edit16_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit14 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit17_Callback(hObject, eventdata, handles)
% hObject    handle to edit13 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit13 as text
%        str2double(get(hObject,'String')) returns contents of edit13 as a
double
```

```
% --- Executes during object creation, after setting all properties.
function edit17_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit13 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
c = str2num(get(handles.edit1,'String'));
gpercent = str2num(get(handles.edit2,'String'));
glpercent = str2num(get(handles.edit3,'String'));
ta = str2num(get(handles.edit4,'String'));
vp = str2num(get(handles.edit5,'String'));
vlpercent = str2num(get(handles.edit6,'String'));
vspercent = str2num(get(handles.edit7,'String'));
mu = str2num(get(handles.edit8,'String'));
rho = str2num(get(handles.edit9,'String'));
E = str2num(get(handles.edit10,'String'));
n = str2num(get(handles.edit11,'String'));
fs = str2num(get(handles.edit12,'String'));
vmfrom = str2num(get(handles.edit13,'String'));
vminterval = str2num(get(handles.edit14,'String'));
vmto = str2num(get(handles.edit15,'String'));

[y1, y2, y3, comparison, which_method] = compare2 (c, gpercent, glpercent,
ta, vp, vlpercent, vspercent, E, n, fs, vmto)

% x=vmfrom:vminterval:vmto;
% figure
% plot(x, y1,x,y2,x,y3)
% legend('Shortening','Lengthening','Accommodating PT')
% xlabel('Volume (vehicle per approach')
% ylabel('Delay caused by adding PT (veh-sec)')
% y=myfunction(x);
% plot(handles.axes1,x, y1,x,y2,x,y3);
% legend('Shortening','Lengthening','Accommodating PT')
% xlabel('Volume (vehicle per approach)')
% ylabel('Delay caused by adding PT (veh-sec)')
% title ({'COMPARISON OF PEDESTRIAN TIMING'; 'ACCOMMODATION METHODS'})
% figure(1);
% plot(x,y);


% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
c = str2num(get(handles.edit1,'String'));
gpercent = str2num(get(handles.edit2,'String'));
glpercent = str2num(get(handles.edit3,'String'));
ta = str2num(get(handles.edit4,'String'));
vp = str2num(get(handles.edit5,'String'));
vlpercent = str2num(get(handles.edit6,'String'));
vspercent = str2num(get(handles.edit7,'String'));
mu = str2num(get(handles.edit8,'String'));
rho = str2num(get(handles.edit9,'String'));
E = str2num(get(handles.edit10,'String'));
n = str2num(get(handles.edit11,'String'));
fs = str2num(get(handles.edit12,'String'));
vmfrom = str2num(get(handles.edit13,'String'));
vminterval = str2num(get(handles.edit14,'String'));
vmto = str2num(get(handles.edit15,'String'));

[y1, comparison, which_method] = compare4 (gpercent, glpercent, vlpercent,
vspercent, mu, rho, E, n, fs, vmfrom, vminterval, vmto)
```