THE DESIGN OF A PC SOFTWARE PACKAGE TO DETERMINE SPEED, SKIDDING TIME, SKIDDING DISTANCE AND DRAG FACTOR OF VEHICLES IMPLEMENTED IN TURBO C FOR MS-DOS V 5.0.

A Thesis Presented to
The Faculty of the College of Engineering and Technology
Ohio University

In Partial Fulfillment
of the Requirement for the Degree
Master of Science

by
Nuruddin S. Abdullah

March, 1994

OHIO UNIVERSITY
LIBRARY
In the name of Allah (God), the Beneficent, the Merciful

1. By the declining day,

2. Lo! man is in a state of loss

3. Save those who believe and do good works, and exhort one another to truth and exhort one another to endurance

(Quran 103, 1-3)
Dedicated to my parents
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Acknowledgements</th>
<th>i</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>ii</td>
</tr>
<tr>
<td>List of Figures</td>
<td>iv</td>
</tr>
<tr>
<td>List of Tables</td>
<td>vii</td>
</tr>
<tr>
<td><strong>CHAPTER I: INTRODUCTION</strong></td>
<td>1</td>
</tr>
<tr>
<td><strong>CHAPTER 2: Road-Vehicle-Driver Interaction: Design Considerations</strong></td>
<td>6</td>
</tr>
<tr>
<td>2.1 Stopping / Skidding Distance</td>
<td>6</td>
</tr>
<tr>
<td>2.2 Perception-Reaction Time</td>
<td>7</td>
</tr>
<tr>
<td>2.3 Skidmarks and Braking Distance</td>
<td>14</td>
</tr>
<tr>
<td>2.4 Coefficient of Friction</td>
<td>22</td>
</tr>
<tr>
<td>2.5 Weight distribution on wheels and Total Drag factor</td>
<td>28</td>
</tr>
<tr>
<td>2.6 Computation of the Total Drag Factor</td>
<td>34</td>
</tr>
<tr>
<td>2.7 Speed / Velocity of skid</td>
<td>42</td>
</tr>
<tr>
<td><strong>CHAPTER 3: VEHICLE DYNAMICS AND LAWS OF MOTION</strong></td>
<td>46</td>
</tr>
<tr>
<td>3.1 Braking dynamics on level road surface</td>
<td>46</td>
</tr>
<tr>
<td>3.2 Braking dynamics on grades (uphill and downhill)</td>
<td>51</td>
</tr>
<tr>
<td>3.3 Skidding on different road surfaces</td>
<td>53</td>
</tr>
<tr>
<td>3.4 Skidding of vehicles with trailers</td>
<td>58</td>
</tr>
</tbody>
</table>
CHAPTER 4: ACCIDENT RECONSTRUCTION PROGRAMS 63

4.1 Accident Reconstruction Programs 63

4.1.1 CRASH-Calspan Reconstruction Accident Speed on the Highway 63

4.1.2 EES-ARM: Equivalent Energy Speed Accident Reconstruction 64

4.1.3 HVOSM - Highway Object Simulation Model 65

4.1.4 IMPAC-Impact Momentum of a Planar Angled Collision 66

4.1.5 SMAC- Simulation Model of Automobile Collision 67

4.1.6 TBS- Tractor Braking and Steering Simulation 67

4.1.7 VTS- Vehicle Trajectory Simulation 68

4.2. Skid Prevention and Counter Measures for Skid Reduction 69

4.2.1 Brake Distribution 70

4.2.2 "Pumping" the Brakes 71

4.2.3 Anti-lock Brake System 71

CHAPTER 5: GUIDELINES FOR SOFTWARE DESIGN 74

5.1. Interactive Programming 74

5.2. Software Support 76

5.3. Displays 77

CHAPTER 6: SOFTWARE SPECIFICATIONS 78

Assumptions and Requirements 78

6.1. Assumptions for the Model 78

6.2. Requirements for the Model 79
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.2.1 User Requirements</td>
<td>79</td>
</tr>
<tr>
<td>6.2.2 Hardware Requirements</td>
<td>81</td>
</tr>
<tr>
<td>6.2.3 Software Requirements</td>
<td>82</td>
</tr>
<tr>
<td>6.2.4 Design Requirements</td>
<td>84</td>
</tr>
</tbody>
</table>

CHAPTER 7: Structure of the Software Package  | 86   |
| 7.1 'Speed' Function Prototype option                                  | 88   |
| 7.2 'Skid_Distance' Function Prototype option                          | 99   |
| 7.3 'Skid_Time' Function Prototype option                              | 101  |
| 7.4 'Coeff_Friction Function Prototype option                          | 103  |
| 7.5 'Calculator' Function Prototype option                             | 105  |
| 7.6 Verification and Validation of the Software                        | 108  |
| 7.6.1 Example 1. Measurement of the Coefficient of Friction            | 108  |
| 7.6.2 Example 2. Calculating the initial speed just before wheels lock | 110  |
| 7.6.3 Example 3. Calculating the Stopping (Braking) Distance           | 111  |
| 7.6.4 Example 4. Calculating the Time taken to produce skidmarks       | 113  |
| 7.7 User Test and Evaluation of software                               | 115  |
| 7.8 Program Documentation and a User's Guide                           | 169  |

CHAPTER 8: CONCLUSIONS AND RECOMMENDATIONS  | 170  |

BIBLIOGRAPHY  | 173  |
APPENDICES

A. Program Listing  
B. Completed questionnaire forms of comments from users  
c. Index of Symbols used in the Software  
D. Index of Functions used in the Software  
E. Derivation of Equations and list of formulas for various conditions  
F. User’s Guide to the Software Package
ACKNOWLEDGEMENTS

I would like to take this moment to render my sincere thanks and appreciation to the people who have supported me throughout my course and my thesis work. My first thanks goes to Dr. Helmut T. Zwahlen under whose coaching and guidance I have been able to get to this point. As my academic advisor as well as my thesis advisor, Dr. H.T. Zwahlen gave me the inspiration and hope in this program and was instrumental in my undertaken this project. Furthermore, I had the opportunity to work under him as a Graduate/Research Assistant. I have without any doubt acquired some experience working with him in many areas, especially in Human Factors. I sincerely thank him for everything. I would like also to express my love and affection to my parents for their patience and understanding. Even though my father is not around to see my achievement, I ask Allah (God) to have mercy on him. It is also a moment to express my sincere love to my mother - Meimunatu and my step mother - Ayesha for their moral support and prayers which has kept me strong and motivated throughout my course and I pray to God to enable me to use my success to help them. My love to the rest of my family, brothers and sisters and friends. I thank the Islamic Center here in Athens for creating a perfect environment for me to study and worship God during my stay here and thank all the brothers and sisters for their assistance and company. My special thanks to Abbas, Ziad, Murali and Baksh for their immense help. I sincerely thank Tom Schnell for taken the time to go through the software and give me valuable ideas. Lastly, I say "All Praise Belongs to Allah (God), the Most Benevolent, Most Merciful".
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Brake reaction times from a study of 321 drivers</td>
<td>10</td>
</tr>
<tr>
<td>2.2a</td>
<td>Median driver perception-reaction times</td>
<td>12</td>
</tr>
<tr>
<td>2.2b</td>
<td>85th percentile driver perception-reaction times</td>
<td>13</td>
</tr>
<tr>
<td>2.3</td>
<td>Skidmarks made by a four-wheel vehicle braking in which the rear wheels skidmarks are superimposed on the front wheels skidmarks</td>
<td>16</td>
</tr>
<tr>
<td>2.4</td>
<td>Four skidmarks made by the sudden braking of a four-wheel vehicle in the simultaneous sliding of all four wheels</td>
<td>17</td>
</tr>
<tr>
<td>2.5</td>
<td>Road-Tire interface forces</td>
<td>23</td>
</tr>
<tr>
<td>2.6a</td>
<td>Variation in coefficient of friction with vehicular speed under various tire and pavement conditions</td>
<td>27</td>
</tr>
<tr>
<td>2.6b</td>
<td>Variation in coefficient of friction with speed on pavements</td>
<td>28</td>
</tr>
<tr>
<td>2.7</td>
<td>Diagram showing incomplete skidmarks of a four-wheel vehicle skidding on one surface</td>
<td>33</td>
</tr>
<tr>
<td>2.8</td>
<td>Diagram showing complete skidmarks of a four-wheel vehicle skidding on two adjacent surfaces</td>
<td>35</td>
</tr>
<tr>
<td>2.9</td>
<td>Relationships involved for determining the height of the center of mass</td>
<td>38</td>
</tr>
<tr>
<td>2.10</td>
<td>Relationships involved for determining the longitudinal position of the center of mass</td>
<td>40</td>
</tr>
<tr>
<td>4.1</td>
<td>Variation of Braking friction coefficient with per cent slip</td>
<td>72</td>
</tr>
<tr>
<td>7.1</td>
<td>Popup window of the main menu</td>
<td>86</td>
</tr>
<tr>
<td>7.2</td>
<td>Flowchart of the software skeleton</td>
<td>87</td>
</tr>
<tr>
<td>7.3</td>
<td>Flowchart of the Speed option function</td>
<td>89</td>
</tr>
<tr>
<td>7.4</td>
<td>Popup windows of the sub menus</td>
<td>90</td>
</tr>
</tbody>
</table>
7.5 Initial speed function flowchart
7.6 Final speed function flowchart
7.7 Complete skidmarks made on two surfaces in which the rear wheels skidmarks are superimposed on the front wheels skidmarks
7.8 Figures designed to show how the front and rear axles drags can be different in a four-wheel vehicle skidding on two adjacent surfaces
7.9 Complete skidmarks made on three surfaces in which the rear wheels skidmarks are superimposed on the front wheels skidmarks
7.10 Skidding distance function flowchart
7.11 Skidding Time function flowchart
7.12 Friction coefficient function flowchart
7.13 Calculator option flowchart
7.14 Traffic Template Nomograph
7.15 Graph of speed as a function of braking distance on two surfaces
7.16 Graph of speed as a function of braking time on two surfaces
7.17 Graph of time as a function of braking distance on two surfaces
7.18 Graph of speed as a function of braking time on three surfaces
7.19 Graph of speed as a function of braking distance on three surfaces
7.20 Graph of time as a function of braking distance on three surfaces
7.21 Graph of braking time as a function of speed
7.22 Graph of braking time as a function of braking distance
7.23 Graph of speed as a function of braking distance 139
7.24 Graph of braking distance as a function of speed of an automobile slowing to a speed of 30 mph at the end of braking 143
7.25 Graph of braking time as a function of speed of an automobile slowing to a speed of 30 mph at the end of braking 145
7.26 Graph of speed as a function of braking distance of an automobile slowing to a speed of 15 mph on two different surfaces 149
7.27 Graph of speed as a function of braking time of an automobile slowing to a speed of 15 mph at the end of braking 151
7.28 Graph of braking time as function of braking distance of an automobile slowing to a speed of 15 mph at the end of braking 153
7.29 Graph of speed as a function of braking distance (A case in which only the rear wheels provided the braking to stop the vehicle) 159
7.30 Graph of speed as a function of braking time of an automobile braking to a complete stop with only the rear wheels braking 161
7.31 Graph of braking time as function of braking distance of an automobile braking to a complete stop with only the rear wheels 163
**List of Tables**

<table>
<thead>
<tr>
<th>TABLE</th>
<th>DESCRIPTION</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Stopping sight distance (wet pavements)</td>
<td>29</td>
</tr>
<tr>
<td>2.2</td>
<td>Effect of grade on stopping sight distance (wet conditions)</td>
<td>31</td>
</tr>
<tr>
<td>7.1</td>
<td>Summary of comments from users who conducted trial runs of the software</td>
<td>114</td>
</tr>
<tr>
<td>7.2</td>
<td>Speed versus Braking distance on two surfaces (two-surface skidding)</td>
<td>120</td>
</tr>
<tr>
<td>7.3</td>
<td>Speed versus Braking time in a skidding involving two surfaces</td>
<td>122</td>
</tr>
<tr>
<td>7.4</td>
<td>Braking time versus braking distance on two surfaces</td>
<td>124</td>
</tr>
<tr>
<td>7.5</td>
<td>Speed versus Braking time in a skidding involving three surfaces</td>
<td>128</td>
</tr>
<tr>
<td>7.6</td>
<td>Speed versus Braking distance in a skidding involving three surfaces</td>
<td>130</td>
</tr>
<tr>
<td>7.7</td>
<td>Braking time versus braking distance on three surfaces</td>
<td>132</td>
</tr>
<tr>
<td>7.8</td>
<td>Braking time versus speed</td>
<td>136</td>
</tr>
<tr>
<td>7.9</td>
<td>Braking time versus braking distance</td>
<td>138</td>
</tr>
<tr>
<td>7.10</td>
<td>Speed versus braking distance</td>
<td>140</td>
</tr>
<tr>
<td>7.11</td>
<td>Braking distance versus speed decreasing to 30 mph</td>
<td>144</td>
</tr>
<tr>
<td>7.12</td>
<td>Braking time versus speed decreasing to 30 mph</td>
<td>146</td>
</tr>
<tr>
<td>7.13</td>
<td>Braking distance versus Speed when braking on two surfaces</td>
<td>150</td>
</tr>
<tr>
<td>7.14</td>
<td>Braking speed versus braking time during a two-surface skidding</td>
<td>152</td>
</tr>
<tr>
<td>7.15</td>
<td>Braking time versus braking distance during which speed decreases to 15 mph at the end of braking</td>
<td>154</td>
</tr>
</tbody>
</table>
7.16 Speed versus braking distance through which only the rear wheels provided the braking

7.17 Speed versus braking time with only the rear wheels braking

7.18 Braking time versus braking distance in which only the rear wheels were locked
CHAPTER 1

INTRODUCTION:

An automobile often skids during emergency braking or during turns on an icy curved road. If the automobile's center of gravity is very high above the ground and it is driven too fast through a tight curve, it may roll over rather than skid. Although it is generally acknowledged that poor "road grip" is the primary variable promoting the onset of skidding, the role of individual tire-vehicle-system elements in the so-called skidding process remains to be established in a definitive quantitative manner.

The word "skid" as defined in the Webster's Collegiate Dictionary is: 1) "to slide without rotating (as a wheel held from turning while a vehicle moves onward)", 2) "to fail to grip the roadway, specifically to slip sideways on the road." The first definition seems to apply to a braking maneuver in which the wheels are not rotating, i.e., the wheels are locked. The second definition suggests a cornering maneuver in which vehicle is side slipping relative to the roadway. These two definitions suggest precisely the meaning that the word "skid" is intended to convey, namely an inability of the driver to exercise control. If wheels are locked during braking, the vehicle may still be stopped in a successful manner. However, the driver has no control of the vehicle’s trajectory. Similarly, a failure to grip the roadway, implies that a driver cannot exercise steering control so as to maintain his vehicle in a given lane.
In a report presented during the First International Skid Prevention Conference, the driver was recognized as a dominant factor in skidding accidents [1]. An analysis of the driver-vehicle highway relationship in skidding accidents reveals a number of facts misunderstood by the driver. Some of these facts are: 1) The friction between the tires and road which may be greatly reduced when the road is wet, which increases the vehicle stopping distance. 2) High vehicle speeds reduce the friction between the road and tires, especially on a wet road surface. 3) The slipperiness of a wet road surface cannot be judged by a motorist simply by looking at it. 4) The slightest swerve, brake application, or acceleration can trigger a skid on a wet or icy road surface, especially at higher vehicle speeds. Unevenly worn or bald tires may cause the vehicle to skid, which can lead to a loss of control especially on wet roads. Skidding is also especially likely to occur at curves, near intersections, on steep hills, and in roundabout. A driver operating his/her vehicle in a skid-hazard zone must be aware that there are a number of maneuvers that require a minimum amount of friction between the road surface and the tires. Such maneuvers are, for example, acceleration, deceleration, and cornering. Of these basic maneuvers, acceleration imposes the least frictional demand since drivers try to achieve the maximum level of acceleration of which the vehicle is capable. Cornering has been one of the principal concerns of skid studies since investigation on skidding commenced [2]. A study by the Texas Highway Department regarding maneuvering along horizontal curves indicated that drivers develop much sharper curvature than design curvature [3].
Vehicle accident reconstruction programs usually involve estimates of the vehicle initial speed and a description of vehicle dynamics by applying the laws of physics. The techniques used to determine the pre-impact vehicle state from post-impact conditions have significantly improved over the past two decades. These improvements can be attributed by a large degree, to the increased interest in the design-related vehicle performance during collisions, the accumulation of dynamic vehicle and occupant behavior data, and the increased availability of computational facilities. Both government and private agencies have funded research institutions for the purpose of developing computer-aided vehicle accident reconstruction methods. These efforts have resulted in a number of interactive programs with which the user can simply and effectively apply the complex mathematical models that represent the vehicle motion from a computer keyboard. Most of the computer programs on skidding for automotive accident reconstruction have been geared towards collision cases involving more than one vehicle. The most commonly used programs are: 1) CRASH-Calspan Reconstruction of Accident Speeds on the Highway, 2) EES-ARM- Equivalent Energy Speed-Accident Reconstruction, 3) HVOSM- Highway Vehicle Object Simulation Model, 4) SMAC-Simulation Model of Automobile Collision, 5) TBS-Tractor braking and Steering simulation, and 6) VTS- Vehicle Trajectory Simulation. The above programs will be discussed again in chapter 4.

There currently exists an urgent need for the integration of human engineering techniques into skid accident reduction programs. For more than two decades, the
highway research community has directed its efforts almost exclusively to studying tire-pavement interactive phenomena. Concurrently, the developments in the automotive industry have been primarily limited to radial tires and anti-skid braking devices. No systematic effort to examine broadly-based causes of skidding accidents has been documented. More sorely neglected is the cause of all skidding accidents - the driver [4].

Many charts and tables have been published showing stopping distances of vehicles from various initial speeds. These charts are based on certain assumptions regarding the reaction time of the driver, the slowing ability of the vehicle, and the pavement and other environmental conditions. Each chart or table is based on a number of different assumptions, thus creating a lot of disparity in their use. In a study on "stopping distances" [5], three main sources of misunderstanding were pointed out in the use of these charts or tables: 1) failure to distinguish between braking distance and total stop-distance; 2) failure to recognize exactly what braking reaction distances are referred to; and 3) failure to recognize exactly what reaction time is referred to.

The objective of this thesis is to develop a user friendly software package to determine the following:

1. The initial (begin) speed given the final (end) speed, the road surface friction coefficient and the skidmarks length.
2. The final speed at a specified skidding distance, given the initial speed and the road surface friction coefficient.
3. The skidding time, given the initial speed, the final speed and road surface
friction coefficient and/or skidmarks length.

4. The road surface friction coefficient, given the initial speed, the final speed and the skidmarks length and/or skidding time.

In addition the software package shall be able to:

5. Solve the above cases (1-4) when skidding occurs uphill or downhill.

6. Solve the above cases (1-4) when skidding occurs on road surfaces with different friction coefficients (maximum 3 different adjacent road surfaces).

7. Plot the graphs of the relationships between the speed, the skidding distance, and skidding time.

8. To generate tables of all graphs plotted in (7).

This package differs from other packages developed in this field in the fact that it is restricted to skidding of vehicles at the instant the brakes are applied such that the wheels remain locked. Having such a program is important for both government and private agencies dealing with transportation and automobile design and also to the police in investigating vehicle accidents. It is a user interactive software package involving questions that automatically appear on the computer screen or printout. Each must be responded to properly by the user.
CHAPTER 2

2. ROAD-VEHICLE-DRIVER INTERACTION - Design considerations:

2.1 Stopping/skidding distance:

Stopping a motor vehicle is performed again and again by every driver in the course of his or her trips. This stopping becomes critical when stopping must be as quick as possible to avoid a collision. It is therefore important for traffic engineers, drivers and other users to have an exact idea of how much distance and/or time it takes to stop an automobile from a given initial speed, or what maximum speed would be allowable in order to be able to stop the vehicle within a given distance or time. Stopping a vehicle consists of a series of operations involving the vehicle, its driver, and the road. This series begins with the reaction to a perceived danger and ends when the vehicle comes to rest.

In order to determine the stopping and/or skidding distance of automobiles, it is important to understand vehicle dynamics. During the stopping process, the kinetic energy of motion of the vehicle or automobile must be converted into friction (heat) energy. When the brakes are applied, the stationary brake shoes expand against the rotating brake drums, and the friction between the shoes and the drums slows down the automobile. Tire-road interface tests indicate that the tire-road coefficient of truck tires, both peak and sliding is significantly less than that of passenger car tires [42]. Any increase in available tire-roadway friction levels will tend to decrease the stopping distance. The faster the automobile is going, the more distance it requires to stop. The
minimum possible stopping distance is about the same if the automobile skids to a stop or rolls to a stop with a braking force just short of skidding. Since the only actual indications of stopping distance in most accident cases are the skid marks left by the braking vehicles, skid mark lengths are considered as the actual measure of stopping distance. The main factors governing stopping distance of an automobile are the following [5]:

1- the speed of the vehicle
2- the grade of the road
3- the percentage of vehicle weight carried on wheels which can be braked (i.e., a vehicle pulling a trailer equipped with brakes as opposed to one pulling a trailer without brakes)
4- the condition of the brakes
5- the road surface friction coefficient.

A vehicle is usually stopped in two operations: 1- driver perception-reaction during which the vehicle loses no speed, 2- vehicle braking which slows the vehicle to a stop.

2.2 Perception- Reaction Time:

The task of driving is a continuous series of visual and audio cues to which the motorist must monitor and respond [42]. The perception of, and reaction to, a particular
cue or stimulus involves four distinct actions on the part of the driver [42].

1- Perception: the recognition or realization that a cue or stimulus exists that requires a response.

2- Intellection or identification: the identification or interpretation of the cue or stimulus.

3- Emotion or decision: the determination of an appropriate response to the cue or stimulus.

4- Volition or reaction: the physical response that results from the decision.

Consider a typical example of a driver approaching a STOP sign. The driver first sees the sign (perception), then recognizes it as a STOP sign (intellection), then decides to STOP (emotion), and finally puts his or her foot on the brake (volition). The total time taken for this sequence of events is referred to as the PIEV time or perception-reaction time. It is a critical parameter in many computational and design analyses. In the case of a driver approaching a STOP sign, the perception-reaction time represents the time between first noticing the sign and depressing the brake pedal. While this is happening, the vehicle continues to move at its initial speed. The perception-reaction time depends on how much thinking is required to decide what cause of action to take. The reaction to a perceive danger can be very simple or complex. For a simple reaction, the driver has decided in advance what to do before something expected happens. For example, he or she has decided to brake if the traffic signal turns yellow or if the brake lights of a car ahead lights up. The decision has already been made, so the time required from
perception to action is short. Also there is a discriminative reaction in which a driver has
to seek additional information to decide what to do. For example, a driver must try to
remember which direction to steer if the rear wheels sideslip, or watch for other signs
when an approaching driver appears to have given the wrong turn signal, or whether the
exit he or she is approaching will or will not take him or her to his or her destination.
 Discriminative reaction may require from a fraction of a second many seconds. Within
that second or seconds, the driver may decide on a complex reaction, which may or may
not be satisfactory. For example, he or she may decide to stop to look at a map or
examine his or her tires before going further. Figure 2.1 illustrates the results of a study
of a simple braking perception-reaction time [6]. The study, conducted by Johansson and
Rumar [10], shows a range of 0.3 to 2.0 seconds for perception-reaction time of 321
drivers used in the study. The Figure 2.1 shows the frequency (or proportion) of drivers
plotted against their reaction times. The first distribution in the figure represents the
assistant’s (the one who was assisting the drivers in the study) reaction times. They were
found to be near-normal distribution with a mean of 0.244 and a standard deviation of
0.016 and a range of 0.20-0.29. From this distribution, it was estimated that the
assistant’s reaction time was not shifting more than 0.05 seconds in either direction. The
median perception-reaction time was 0.66 seconds with 10 percent requiring 1.5 seconds
or longer. The same study compared brake reaction times between anticipated and
unexpected cues by establishing a correction factor to account for the variance in reaction
times for unexpected and expected cues. This approach found that those for unexpected
Figure 2.1 Brake Reaction times from a study of 321 drivers.
Source: Hulbert, "Human factors in Transportation," Transportation and Traffic Engineering Handbook, Institute of Transportation Engineers, Washington, D.C 1982, Fig. 8-4 p-216.
cues were from 0.1 to 0.3 seconds higher than for anticipated cues. Figures 2.2a and 2.2b further emphasizes the difference in reaction times between expected and unexpected cues for median-case drivers and 85th percentile drivers respectively. Information is represented in the Figures 2.2a and 2.2b in bits which is the term used to quantify the amount of information needed to make a decision. Another study [7] gave 0.64 seconds as the average perception-reaction time, with 5 percent of the drivers requiring over 1 second. In a third study [8], reaction time values ranged from 0.4 to 1.7 seconds. Woods [43] noted that any change in perception-reaction time was actually a change in the distance traveled at the design speed. Thus, the effect on stopping sight distance (SSD) was highly speed dependent. Glennon [44] and Farber [45] indicated that for changes in perception-reaction time, the increase in SSD became significant at higher speeds. Hooper and McGee [46] reached the opposite conclusion, contending that at higher speeds, the braking component of stopping sight distance became the dominant factor, even though a significant distance was traveled during the increased perception-reaction time. The American Association of State Highway and Transportation Officials recommends the use of 2.5 seconds for perception-reaction time in computations involving stopping or braking reactions [9]. This has been confirmed as appropriate for most drivers by a number of studies, including most recently, Johansson and Rumar [10] and Olson et al. [11].
Figure 2.2a Median driver perception-reaction times.
Source: A Policy on Geometric Design of Highways and Streets. The American Association of State Highway and Transportation Officials. Washington, DC, 1984, Fig. 11-14, p.43
Figure 2.2b 85th-percentile driver perception-reaction times.

The distance traveled by an automobile during the perception-reaction time is given by:

\[ d_p = 1.4667vt \]  

(2.1)

where

- \( d_p \) = perception-reaction (PIEV) distance, feet
- \( v \) = speed during perception time, mph (constant)
- \( t \) = perception time, seconds

1.4667 = conversion factor (1 mph = 1.4667 feet/sec)

This formula is accurate provided the speed does not increase or decrease while the driver is perceiving and is forming his/her response.

2.3 Skidmarks and Braking Distance:

A skidmark is a friction mark made by a vehicle tire which is not rotating. Skidmarks may vary considerably in appearance depending on the kind of surface they are made on and the condition of that surface. Skidmarks are used in accident investigations as signs of braking, slowing, and loss of speed. Some characteristics of skidmarks include:

1. They follow a relatively straight path.
2. They are generally four marks from a four-wheeled vehicle but they may be two marks, three marks, or only one mark.
3. Marks made by left and right tires are usually equally dark and equally wide.

4. Front-tire skidmarks are usually more prominent than rear-tire skidmarks.

5. Skidmarks nearly always ends abruptly, either when the vehicle stops or where a collision begins.

Four patterns [5] generally encountered in examining the roadway after an accident showing skidmarks are:

1. There are no signs of skidmarks although there may be other kinds of tire marks. This normally happens when there is not sufficient braking to lock wheels or the vehicle is equipped with an antilock braking system (ADS), or that the skidmarks disappeared before they could be observed.

2. Skidmarks of nearly the same length are made by all wheels. Such marks indicate that all wheels were locked suddenly and that the whole vehicle slid (Figure 2.4) or Figure 2.3 if the rear wheel marks are so exactly superimposed on the front wheel marks.

3. There are definite skidmarks from all wheels but they vary considerably in length. This happens when some wheels lock before others due to the way the brakes are applied.

4. Skidmarks made by some, but not all the wheels.
Figure 2.3 Skidmarks made by a four-wheel vehicle braking in which the rear wheels skidmarks are superimposed on the front wheels skidmarks.
Figure 2.4 Four skidmarks made by the sudden braking of a four-wheel vehicle in the simultaneous sliding of all four wheels.
The skid distance in the above cases is the distance moved by the front or rear wheels from the start of the skid to the end of the skid, or the distance moved by the vehicle’s center of gravity.

In highway safety and design, the most critical performance characteristic of vehicles is braking, or deceleration [42]. The time and distance required to stop a vehicle is a primary consideration in virtually every aspect of traffic-system design and operation. Braking performance is related to a number of factors, including the vehicle’s braking system, the type and condition of tires, and the condition and type of roadway surface. How far a vehicle goes while braking to a stop depends on how fast it was moving to begin with and how much slowing drag the brakes and road surface can develop. Two rates of slowing can be specified in most cases: 1) maximum slowing and minimum stopping distance determined by the friction of road (drag factor); and 2) maximum stopping distance permitted by law. In the first case, brake pedal pressure beyond that required to lock wheels produces no shorter stopping distances, assuming that brakes and driver are capable of locking the wheels. Steps required to estimate braking distances are as follows [5]:

1. Specify the speed from which the stop is to be made. This may be the speed limit or a speed at which a driver stated he or she was traveling.

2. Determine the drag factor which represents the rate of slowing. Maximum slowing (minimum braking distance) can be obtained using the values for skidding with wheels locked. Measurements show that the drag factor is not the same for all speeds. It
decreases as the initial speed increases. It varies considerably because of many physical elements such as air pressure of tires, composition of tires, tire tread pattern and depth of tread, type and condition of the pavement surface, and the presence of moisture, mud, snow, or ice. In general, the distance required to decelerate a vehicle from one speed to another (with the use of brakes) is given as

\[
s = \frac{V_i^2 - V_e^2}{30 (f \pm g)}
\]  

(2.2)

where

- \( s \) = braking distance (length od skidmarks), feet
- \( V_i \) = initial speed of vehicle, mph
- \( V_e \) = final speed of vehicle, mph
- \( f \) = coefficient of friction (dimensionless)
- \( g \) = grade - the ratio of vertical height divided by distance for which vertical height was measured. (g is +ive when grade is uphill and -ive when downhill)
- \( 30 \) = conversion factor (see derivation in Chapter 3)

When the braking is on a level road, \( g=0 \), and when the vehicle slide to a stop, the final speed \( (V_e) \) of the vehicle is zero. This formula is accurate provided the drag factor does
not increase or decrease during the braking period. In practice, unfortunately, the drag factor usually does change somewhat especially during long stops from high speeds. The braking distance can also be calculated by the use of a nomograph. This consists of scale readings of such parameters as speed, friction coefficient, braking distance, and time. Given any two of the parameters, the third and fourth can be found. To find the braking distance on the nomograph, locate the desired speed and the selected drag factor and draw a straight line through these points on the scales and extend it to the braking distance scale. The point it cuts the braking distance scale is the value of the braking distance for that speed. The total stopping distance (TSD) is the sum of the perception-reaction distance and the braking distance. That is Equation 2.1 + Equation 2.2. or expressed as

\[
TSD = d_p + s
\]  

\[
TSD = 1.4667 vt + \frac{v_i^2 - v_o^2}{30 (f \pm g)}
\]

where

TSD = total stopping distance, feet

d_p = perception-reaction distance, feet

s = braking distance, feet.
The total distance formula can be used to establish safe stopping distances for motorists.

A cardinal rule of highway design is that alignment must allow drivers to see a distance at least equal to the minimum safe stopping distance for the highway. It is assumed that, in the worst case, a driver encounters an object in his or her lane and evasive action is not possible. Thus, the driver must see the object in time to react and stop the vehicle.

Consider a highway with a design speed of 65 mph and a coefficient of friction given as 0.30. Assuming the highway is level and braking brings the automobile to a complete stop and using the AASHTO recommended reaction time of 2.5 seconds, the safe stopping distance can be computed from the total distance formula as:

\[ TSD = (1.4667 \times 65 \times 2.5) + \frac{65^2 - 0}{30(0.30 + 0)} \]

\[ TSD = 708 \text{ feet}. \]

Thus the highway must be designed such that drivers in either direction have a minimum of 708 ft. of stopping sight distance. Failure to provide this would allow a driver to see a hazard initially at a point too close to allow a safe stop.

The total stopping distance is a product of the driver, the vehicle, and the roadway. Because of this, the total stopping distance of individual drivers and vehicles in various situations does vary considerably. Design values are chosen on the conservative side of observed behavior to provide for reasonable safety of the road-user.
population. It has been argued that the stopping sight distances are often inadequate when drivers must make complex or instantaneous decisions, when information is difficult to perceive, or when unexpected or unusual maneuvers are required. Decision sight distance provides the greater length that drivers need. Decision sight distance, is the distance required for a driver to detect an unexpected or otherwise difficult-to-perceive information source or hazard in a roadway environment that may be visually cluttered, recognize the hazard or its threat potential, select an appropriate speed and path, and initiate and complete the required safety maneuver safely and efficiently [12]. Drivers need decision sight distance whenever there is a likelihood for error in either information reception, decision making, or control actions [13].

2.4 Coefficient of Friction

The primary forces acting on an automobile during a braking effort are: tire normal force, the tire braking force, and the tire lateral force. Figure 2.5 shows how these forces act on the tire of an automobile. The ratio of the tire braking force to the tire normal force represents a measure of the slipperiness of a road or surface. This measure of slipperiness is termed the coefficient of friction. It is also known by other names such as: adhesion factor, drag factor, road surface factor, pavement friction, etc.. The friction between the tire and the roadway limits the maximum braking capability of a vehicle. The friction coefficient depends upon the material and surface geometries of the tire and the roadway, the nature and thickness of any film such as water, sand, or oil present in the contact area, sliding velocity and the operating temperature.
Figure 2.5 Road-Tire Interface Forces
Very slippery surfaces have a low coefficient of friction, while surfaces upon which nothing wants to slide have a high coefficient of friction. In other words, the higher the coefficient of friction, the less slippery the surface is. The coefficient of friction is equal to the drag factor when the vehicle is traveling on a level surface and the application of the brakes locks all the wheels. The drag factor is the horizontal force required to overcome resistance in moving a vehicle or other object along a surface divided by the weight of the vehicle or object. The coefficient of friction is also sometimes called the "skid number." It is usually a decimal fraction, and sometimes multiplied by 100 to express it as a percentage. It can be greater than 1.00 or 100 percent. The higher the coefficient of friction, the more frictional force can be generated, and consequently the shorter the stopping distance is from a given speed. When a wheel is locked and slides, the retarding force on the wheel is the load (weight) on that wheel times the drag factor for that wheel. So when the wheel is not rotating, the drag factor is determined by the sliding coefficient of friction between the tire and the road surface. If all tires do not slide or if they slide on different surfaces, the drag factor for some wheels will be different from that of others. Then the wheels with greater drag factor will produce the retarding force to slow some of the load carried by wheels with lesser drag factors. The matter of total drag factor when individual wheels have different drag factors is complicated by the fact that in slowing, a vehicle’s weight shifts from rear wheels to front wheels. This is because when slowing, the vehicle tends to roll forward on its nose. As a result, a vehicle which has half its weight on the front and half on rear axle when standing may have 0.6 of its weight on the front and 0.4 of its weight on the rear when
braking. This is the main reason for putting stronger brakes on front wheels than on rear wheels. The coefficient of friction can be calculated from the braking distance Equation (2.2) by expressing it as

\[ f = \frac{V_i^2 - V_e^2}{30s} \pm g \]  

(2.5)

where

- \( f \) = coefficient of friction (dimensionless)
- \( s \) = braking distance or length of skidmarks, feet
- \( V_i \) = initial speed of vehicle, mph
- \( V_e \) = final speed of vehicle, mph. (\( V_e = 0 \), if vehicle slides to a stop).
- \( g \) = grade of skid distance (\( g = 0 \) if vehicle slides on a level surface, +ive for uphill skid, -ive for downhill skid).

The actual coefficient of friction can be measured [5], at the scene of an accident by using a spring scale to measure the amount of pull or frictional force (\( F \)) in pounds, required to slide a tire of known weight \( W \) on a road surface, or sliding a vehicle to a stop with locked wheels, that is, a test skid. The frictional force (\( F \)) equals the coefficient of friction (\( f \)) times the weight of the sliding object (\( W \)).
That is

\[ F = f w \]  \hspace{1cm} (2.6)

and the coefficient of friction can be expressed as

\[ f = \frac{F}{w} \]  \hspace{1cm} (2.7)

where

\begin{align*}
F &= \text{frictional force to slow down the vehicle, pounds} \\
f &= \text{coefficient of friction of road surface (dimensionless)} \\
w &= \text{weight of vehicle, pounds}.
\end{align*}

Various studies have been conducted on coefficient of friction for both wet and dry conditions. In one study [14], more than 1000 measurements of forward skidding were made on 32 pavements. The curves 1 to 6 in Figure 2.6A illustrates the friction coefficients found on the various pavements in study [14] and Figure 2.6B summarizes the results of a series of 600 measurements made on modern pavements in Germany [15]. Each road section was tested at 20, 40, 60, and 80 km/hr (12.5, 25, 37.5, and 50 mph) by a locked wheel trailer. The Figure 2.6B does indicate that there is a wide variation between the coefficient of friction for various pavements, which indicates the effect that road surface texture and condition has on the coefficient of friction.
Figure 2.6 Variation in coefficient of friction with vehicular speed. Source: AASHTO-Geometric Design of Highways and Streets. 1984, p. 141
The minimum drag factor for a specify the maximum stopping distance and a given speed (usually 30 feet from 20 mph) is 0.446 [5]. Brakes which cannot produce a drag as high as 0.446 are illegal [5]. The friction coefficient of 0.466 does not apply in all states, and in some states, applies to all types of vehicles. Table 2.1 shows computed distances for wet pavements and for various speeds. The friction coefficient values used in calculating safe stopping distances should allow for worn tires as well as new tires and for nearly all types of treads and tire compositions. The coefficients of friction in Table 2.1 have been selected based on the above criteria.

2.5 Weight Distribution on Wheels and Total Drag Factor

The weight acting on the tires makes a great difference in the force needed to slide a vehicle on a road surface. Even though the coefficient of friction does not depend on the vehicle weight, the weight distribution among the tires is significant for vehicle stability during braking. Each wheel produces a drag which is determined by the part of the vehicle's weight on that wheel and the coefficient of friction. If the weight distribution on all wheels are about the same and the wheels are sliding on the same surface, then the drag factor produced by the wheels is the same as the coefficient of friction of the surface. However, when the surface is not level and the weight distribution is the same or nearly the same on each wheel, its drag factor is the coefficient of friction of the road plus the grade factor if the surface is upgrade, or minus grade factor if the surface is downgrade.
Table 2.1 Stopping sight distance (wet pavements).

<table>
<thead>
<tr>
<th>Design Speed (mph)</th>
<th>Assumed Speed for Condition (mph)</th>
<th>Brake Reaction</th>
<th>Coefficient of Friction on Level</th>
<th>Stopping Sight Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time (sec)</td>
<td>Distance (ft)</td>
<td></td>
<td>Computed(^a)</td>
</tr>
<tr>
<td>20</td>
<td>2.5</td>
<td>73.3-73.3</td>
<td>0.40</td>
<td>106.7-106.7</td>
</tr>
<tr>
<td>25</td>
<td>2.5</td>
<td>88.0-91.7</td>
<td>0.38</td>
<td>138.5-146.5</td>
</tr>
<tr>
<td>30</td>
<td>2.5</td>
<td>102.7-110.0</td>
<td>0.35</td>
<td>177.3-195.7</td>
</tr>
<tr>
<td>35</td>
<td>2.5</td>
<td>117.3-128.3</td>
<td>0.34</td>
<td>217.7-248.4</td>
</tr>
<tr>
<td>40</td>
<td>2.5</td>
<td>132.0-146.7</td>
<td>0.32</td>
<td>267.0-313.3</td>
</tr>
<tr>
<td>45</td>
<td>2.5</td>
<td>146.7-166.0</td>
<td>0.31</td>
<td>318.7-382.7</td>
</tr>
<tr>
<td>50</td>
<td>2.5</td>
<td>161.3-183.3</td>
<td>0.30</td>
<td>376.4-461.1</td>
</tr>
<tr>
<td>55</td>
<td>2.5</td>
<td>176.0-201.7</td>
<td>0.30</td>
<td>432.0-537.8</td>
</tr>
<tr>
<td>60</td>
<td>2.5</td>
<td>190.7-220.0</td>
<td>0.29</td>
<td>501.5-633.8</td>
</tr>
<tr>
<td>65</td>
<td>2.5</td>
<td>201.7-238.3</td>
<td>0.29</td>
<td>549.4-724.0</td>
</tr>
<tr>
<td>70</td>
<td>2.5</td>
<td>212.7-256.7</td>
<td>0.28</td>
<td>613.1-840.0</td>
</tr>
</tbody>
</table>

\(^a\)Different values for the same speed result from using unequal coefficients of friction.
Table 2.2 shows the effect of grade on stopping distance under wet conditions. Different criteria are used in calculating the corrections for upgrade and downgrade because of their different effects on the speed of individual vehicles, particularly large trucks. If the weight distribution is not the same on the wheels on opposite ends of an axle, the drag factor of that axle is the sum of the drag factor on one wheel times the fraction of load on that wheel and the drag factor of the opposite wheel times the fraction of load on that wheel. The total drag factor can be estimated by multiplying the part of the vehicle's weight on each wheel by the drag factor for that wheel and adding the products so obtained.

Consider the following example in which a vehicle has its rear wheels locked and sliding on a surface with a coefficient of friction of 0.70 and its front wheels rolling with a friction of 0.01. Assume the vehicle has 40 percent of its weight on the front wheels and 60 percent on the rear wheels, then the drag factor of the rear wheels will be more than the drag factor for the front wheels. The total drag factor will be

\[
\begin{align*}
\text{Front wheels} & \quad 0.40 \times 0.01 = 0.004 \\
\text{Rear wheels} & \quad 0.60 \times 0.70 = 0.420 \\
\text{Total drag factor} & \quad 0.424
\end{align*}
\]
Table 2.2 Effect of grade on stopping sight distance (wet conditions).

<table>
<thead>
<tr>
<th>Design Speed (mph)</th>
<th>Increase for Downgrades</th>
<th>Decrease for Upgrades</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Correction in Stopping</td>
<td>Assumed Speed for</td>
</tr>
<tr>
<td></td>
<td>Distance (ft) (mph)</td>
<td>Condition (mph)</td>
</tr>
<tr>
<td></td>
<td>3% 6% 9%</td>
<td>3% 6% 9%</td>
</tr>
<tr>
<td>30</td>
<td>10 20 30</td>
<td>28 – 10 20</td>
</tr>
<tr>
<td>40</td>
<td>20 40 70</td>
<td>36 10 20 30</td>
</tr>
<tr>
<td>50</td>
<td>30 70 –</td>
<td>44 20 30 –</td>
</tr>
<tr>
<td>60</td>
<td>50 110 –</td>
<td>52 30 50 –</td>
</tr>
<tr>
<td>65</td>
<td>60 130 –</td>
<td>55 30 60 –</td>
</tr>
<tr>
<td>70</td>
<td>70 160 –</td>
<td>58 40 70 –</td>
</tr>
</tbody>
</table>
If all the wheels of a vehicle do not slide or slides on different surfaces, the drag factor for some wheels will be different from that of others. In such a case, the software will attempt to calculate the total drag factor of the vehicle if some information can be provided. Two cases will be presented to demonstrate how the software handles cases of unequal drag on the wheels.

1. Figure 2.7 represents a case of incomplete skidding on a level portland cement concrete. The vehicle started with a single skidmark from A to B, then made two skidmarks with the rear wheels from B to C, then three skidmarks from the rear wheels and one front wheel from C to D and finally complete skidmarks involving all four wheels from D to E. The software package does not solve the example in Figure 2.7 as a whole unit but can be used to find the total drag factor at each phase of the skidding, if each phase is treated as a single and independent case. For example, in the skidding from B to C, the rear wheels made two skidmarks. The software, after getting user input that not all wheels are locked, will request for the percentage of weight carried by the sliding wheels (in this case the rear wheels). The information is then used to compute the rear axle drag and the front axle drag. If the two drag factors are not equal, the software will attempt to compute the total drag factor for the phase B to C provided the user provides some information which are needed for this computation, otherwise, the user is informed that it is not possible to compute the total drag factor. The computation of the total drag factor will be presented in the next section.
Figure 2.7 Diagram showing incomplete skidmarks of a four-wheel vehicle skidding on one surface.
2. The second case in which the software attempts to compute total drag factor is demonstrated by the example in Figure 2.8, in which a vehicle is skidding on two surfaces, in this case, level portland cement concrete and loose gravel. In Figure 2.8, the vehicle has all four wheels entirely on the cement concrete from A to B, then one wheel on the gravel from B to C, then two wheels on the gravel from D to E, then three wheels on the gravel, and finally all four wheels on the gravel from E to F. For phases A to B and D to E in which all the four wheels are entirely on the cement concrete and gravel respectively, the drag factor for phase A to B is equal to the friction coefficient of the cement concrete. Similarly, the drag factor for phase D to E is equal to the friction coefficient of the gravel. The other phases in which some wheels are on the cement concrete and some on the gravel will have different drag factors and hence the need to calculate the total drag factors for those phases. The following section explains how the total drag factor is computed.

2.6 Computation of the Total Drag factor

In this section all the information and formulas pertaining to the computation of the total drag factor are taken directly and some indirectly from Stannard [48]. The calculation of the total drag factor under the function prototype 'Combined drag' in the software requires the following input information:

1. length of the wheel base of the vehicle, L feet
2. weight of the vehicle, w lbs
Figure 2.8 Diagram showing complete skidmarks of a four-wheel vehicle skidding on two adjacent surfaces.
3. height of the center of mass of the vehicle as a fraction of the wheel base, \( x \) feet.

4. location of the center of mass to the rear of the front axle as a fraction of wheel base length, \( y_f L \) feet.

5. drag factor on each axle, \( f_r \) and \( f_t \).

Since published data on vehicle dimension rarely include the location of the center of mass, information available can lead to the calculation of this center of mass [48]. These available information are:

1. static weight on front (\( w_f \)) and rear (\( w_r \)) axles or fraction (percent) of weight on either front or front axles.
2. length of vehicle wheel base (distance from the front axle to the rear axle).
3. radius of wheel with tire, \( R \) feet
4. weight on one axle, \( w_h \), when the other axle is lifted to a height, \( h \)
5. height, \( h \) to which the axle is lifted for weighing to locate the center of mass.

With the above information, the location of the center of mass can be estimated by the formula:

\[
x = \left( \frac{R}{L} \right) + \left( \frac{w_h - w_t}{h} \right) \frac{\sqrt{L^2 - h^2}}{h_w}
\]
or

\[ x = \frac{(w_h - w_f)}{w \tan b} + \frac{R}{L} \]

where

- \( x \): height of the center of mass of the vehicle, feet
- \( R \): radius of the vehicle wheels with tires, feet
- \( L \): length of the wheel base of the vehicle, feet
- \( w_h \): weight of one axle when the other axle is lifted \( h \) ft, lbs
- \( w_f \): weight carried by the front wheels, lbs
- \( w \): weight of the vehicle, lbs
- \( h \): height to which an axle is lifted, feet
- \( \tan b \): angle of tilt when one axle is lifted \( h \) feet.

Figure 2.9 shows a diagram of a vehicle hoisted for weighing to determine the height of the center of mass. By hoisting the vehicle from one axle, in this case, the rear axle, the quantity \( w_h \) is measured.
Figure 2.9  Relationship involved in equation for determining the height of the center of mass of a vehicle.
Source: J.B. Stannard, "Estimating Stopping Distances and Time for Motor Vehicle". Traffic Institute, Northwestern University, Evanston, Il. 1977, p.75
The center of mass (center of gravity) is a point at which the entire mass of a body may be considered concentrated, or the point at which a body will balance fore and aft and from side to side. The longitudinal (fore and aft) position of the center of mass can be calculated by applying the principle of moments to the static weight on the front and rear axles and the wheel base length. Figure 2.10 shows the relationships involved in equation for determining longitudinal position of the center of mass of a vehicle. Applying moments at \( P_f \) both clockwise and counterclockwise in Figure 2.10 gives:

\[
\text{Clockwise moment} = w y_f L
\]

\[
\text{Counterclockwise moment} = w y_r L
\]

Since moments about a point are equal,

\[
w y_f L = w y_r L
\]

Dividing each side by \( L \) and solving for \( y_f \), gives:

\[
y_f = w y_f / w
\]

Likewise, taking moments about \( P_r \),

\[
y_r = w y_r / w
\]

That is, the proportion (decimal fraction) of the wheel base between the center of mass and either axle is equal to the proportion of the vehicle's weight on that axle. If the weight distribution of the vehicle is given as percent of total weight on either axle, the center of mass is that percent of the wheel base away from the other axle.
Figure 2.10 Relationships involved in equation for determining the longitudinal position of the center of mass of a vehicle.
The actual distance of the center of mass behind the front axle is \( y_f L \) and the actual distance of the center of mass ahead of the rear axle is \( y_r L \). Since the wheel base length is equal to the distance from the center of mass to the front axle plus the distance to the rear axle:

\[
L = y_f L + y_r L
\]

or \( y_f L = L - y_r L \)

If all the information needed are supplied by the user, the software will compute the height of the center of mass, position of the center of mass and uses that information to compute the total drag by applying the formula:

\[
f = f_f - y_f (f_f - f_r) / (1 - x (f_f - f_r))
\]

where

- \( f \) = total drag factor (dimensionless)
- \( f_f \) = front axle (wheels) drag factor (dimensionless)
- \( f_r \) = rear axle (wheels) drag factor (dimensionless)
- \( y_f \) = location of the center of mass to the rear of the front axle, feet
- \( x \) = height of the center of mass of the vehicle, feet

Note: As mentioned at the beginning of this section, all information and formulas in this section are taken from Stannard [48].
2.7 Speed /Velocity

Speed or velocity is the distance divided by time or the rate of motion. For most purposes speed and velocity mean the same thing: rate of travel. (Technically, velocity includes direction as well as the rate of motion). Three types of speed come into play when a vehicle in motion is trying to stop.

1. Initial speed: This is the rate at which a vehicle (or other object) is moving before skidding starts. It is the speed during which a driver uses his or her perception-reaction time to decide what cause of action to take to avoid a possible collision or danger.

Accident investigators often make use of measured skid-marks to estimate vehicle speeds before an accident. Such measurements are used together with the knowledge of the friction coefficient, estimates of collision speed, and the braking distance formula to approximate the initial speed of a vehicle.

2. Slow-to stop speed (Skid speed): This is the rate at which a vehicle is sliding before it comes to a stop or before it collides with something. Before the brakes are applied, the vehicle is traveling at a certain initial speed mentioned above. At the instant the brakes are applied, the initial speed starts reducing until the vehicle comes to a complete stop or hits something.
3. **End (final) speed**: This is the rate at which the vehicle is moving at the end of its sliding motion. If the vehicle comes to complete stop at the end of sliding, this final speed is zero. However, if it crashes into something, rolls over, falls or flips, then its final speed at the end of skidding is not zero but some value to be determined.

One of the laws of motion states that: a vehicle in motion continues to move until its energy of motion (kinetic energy) is exhausted in one way or another. Part of the vehicle's energy may be dissipated as friction heat in the slowing process and the remainder in a crash damage or otherwise at the end of slowing by braking. The combined energy at the beginning of slowing is the sum of the energy required to slow to a stop in a specified distance and the energy remaining at the end of slowing. It can be proven that the speed at the beginning of slowing equals the square root of the sum of the speed required to slow to a stop in a specified distance squared plus the speed at the end of slowing squared. This can be expressed as:

\[
V_i^2 = V_s^2 + V_o^2 \tag{2.8}
\]

\[
V_i = \sqrt{V_s^2 + V_o^2} \tag{2.9}
\]

where

\( V_i \) = combined speed at start of skid or initial speed, mph.
\[ V_s = \text{slide to stop speed, mph. (speed derived from slide distance)} \]

\[ V_f = \text{final or speed at the end of skidding, mph.} \]

The slide to stop speed can be calculated by using the braking distance formula. If the speed at beginning of slowing is known, the speed at any distance thereafter can be calculated.

Consider the following example. A vehicle hits a bridge abutment at a speed estimated by investigators as 20 mph, leaving first 100 feet skid marks on the pavement \( (f=0.35) \) and then additional skid marks of 200 feet on the gravel shoulder approaching the abutment \( (f=0.50) \). The grade is level. What is the initial speed of the vehicle?

Two braking distances are known: 100 feet on the pavement and 200 feet on the shoulder, each of which has an initial and a final speed. Since the final speed is given as 20 mph, the speed at which the vehicle was going when it started to skid on the gravel can be determined as:

\[
200 = \frac{V_i^2 - 20^2}{30(0.50+0)}
\]

\[
3000 + 400 = V_i^2
\]
\[ V_i^2 = 3400 \]

\[ V_i = 58.3 \text{ mph} \]

This is the speed at the beginning of the gravel skid or the speed at the end of the pavement skid. The speed at the beginning of the pavement skid is

\[ 100 = \frac{V_i^2 - 58.3^2}{30(0.35 + 0)} \]

\[ 3400 + 1050 = V_i^2 \]

\[ V_i^2 = 4450 \]

\[ V_i = 66.7 \text{ mph} \]

This speed information, along with other aspects of the accident investigation, can be used to help determine whether excessive speed was a contributory cause of the accident.
CHAPTER 3

3. VEHICLE DYNAMICS AND LAWS OF MOTION:

The primary forces acting on a vehicle during a braking effort are: the tire normal force, the tire braking force, and the tire lateral force. Refer to Figure 2.4 to see how these forces act on the automobile tire. These forces occur at each road-tire interface and determine the path of a vehicle during the braking effort. The tire normal force is a reaction to the vehicle weight distribution, dynamically varying during a braking maneuver. The tire braking force is a reaction to the braking torque on the wheels. The lateral tire force is a reaction to any lateral motion of the vehicle. The ratio of the tire braking force to the tire normal force gives the coefficient of friction of the surface.

3.1 Braking/stopping dynamics on level surfaces

As noted in chapter 2, stopping is accomplished by changing motion energy into frictional (heat) energy. The conversion is done by the frictional force acting between the tires and the road. The frictional force always acts to slow down the vehicle. The more frictional force exerted, the more rapidly the vehicle will slow down. One natural law says that force is equal to the mass times acceleration. This is expressed as

$$F = ma$$

(3.1)
where

\[ F = \text{force, lbs} \]

\[ m = \text{mass, lbs/feet/sec}^2 \text{ (slugs)} \]

\[ a = \text{deceleration, feet/sec}^2 \]

The mass of anything is equal to its weight \( w \) in pounds divided by acceleration due to gravity. This is expressed as:

\[ m = \frac{w}{g} \]  \hspace{1cm} (3.2)

where

\[ m = \text{mass, lbs/feet/sec}^2 \text{ (slugs)} \]

\[ w = \text{weight, lbs} \]

\[ g = \text{acceleration due to gravity} = 32.2 \text{ ft/sec/sec}. \]

Equating equations 2.6 and 3.1 gives

\[ f_w = ma \]  \hspace{1cm} (3.3)

Substituting equation 3.2 in equation 3.3 gives

\[ f_w = \left( \frac{w}{g} \right) a \]  \hspace{1cm} (3.4)
Dividing both sides of equation 3.4 by \( w \) and rewriting it with respect to \( a \)

\[ a = fg \]  \hspace{1cm} (3.5)

where

\( a = \) deceleration of vehicle, feet/sec\(^2\)

\( f = \) friction coefficient

\( g = \) acceleration due to gravity, 32.2 feet/sec\(^2\).

Equation 3.5 says that the deceleration of the vehicle depends only on the coefficient of friction, and is independent of the vehicle weight. Another natural law of motion says that the distance \( s \) traveled equals one-half the deceleration \( a \) times the time \( t \) squared, or is expressed as:

\[ s = \frac{1}{2} at^2 \]  \hspace{1cm} (3.6)

\[ v = at \]  \hspace{1cm} (3.7)

Solving Equation 3.7 for \( t \), we get
and by substituting Equation 3.8 for $t$ in Equation 3.6, we obtain

$$s = \frac{1}{2} a \left( \frac{V}{a} \right)^2$$  \hspace{1cm} (3.9)

or,

$$s = \frac{1}{2} \frac{V^2}{a}$$  \hspace{1cm} (3.10)

Note: Equation 3.10 is true only if $a$ is assumed to be constant.

Substituting Equation 3.5 for $a$ in Equation 3.10,

$$s = \frac{1}{2} \frac{V^2}{\frac{1}{2} fg}$$  \hspace{1cm} (3.11)

Valid only for level or horizontal skid (no grade).

If the stopping distance $s$ in Equation 3.11 is required in feet, and $V$ is given in miles per hour, the units have to be converted into the same units.
1 mph = 5280 feet per hour

1 mph = \( \frac{5280}{3600} \) feet per second

1 mph = 1.4667 feet per second \hspace{1cm} (3.12)

Using the conversion factor in equation 3.12, and 32.2 as gravity \( g \), Equation 3.11 becomes:

\[
s = \frac{1}{2} (1.4667)^2 \frac{V^2}{(f)(32.2)}
\]  

(3.13)

Rewriting equation 3.13, gives

\[
s = \frac{V^2}{30f}
\]  

(3.14)

where

\( s = \) stopping distance, ft.

\( V = \) velocity, mph.
f = coefficient of friction
30 = conversion factor

Equations valid for horizontal (no grade) skid only and end speed \((V_e) = 0\) mph.

3.2 Braking/stoping dynamics on grades.

The stopping distance formula of Equation 3.14 assumes that a vehicle is brought to a complete stop by the application of the brakes, so its final velocity \((V_e)\) is zero. It also assumes that the vehicle is traveling on a level road in which case the grade of the road is zero. If the vehicle collides with something, its final velocity will not be zero. Also if it is traveling on a grade (uphill, downhill), Equation 3.14 will be modified as follows:

\[
s = \frac{V_i^2 - V_e^2}{30(f+g)} \quad \text{(uphill skid)} \tag{3.15}
\]

\[
s = \frac{V_i^2 - V_e^2}{30(f-g)} \quad \text{(downhill skid)} \tag{3.16}
\]

where

\[
s = \text{stopping distance, feet.}
\]
\[ V_i = \text{initial velocity of vehicle, mph} \]
\[ V_e = \text{final velocity of vehicle, mph} \]
\[ f = \text{coefficient of friction} \]
\[ g = \text{grade (} +g \text{ if uphill and} -g \text{ if downhill)} \]
\[ 30 = \text{conversion factor} \]

From Equations 3.15 and 3.16, the speed at the start of skidding can be computed as:

\[ V_i^2 = s(30(f \pm g)) + V_e^2 \]

\[ V_i = \sqrt{30s(f + g) + V_e^2} \quad \text{(uphill skid)} \quad (3.17) \]

\[ V_i = \sqrt{30s(f - g) + V_e^2} \quad \text{(downhill skid)} \quad (3.18) \]

\[ V_i^2 = 30sf \pm 30sg + V_e^2 \]

The friction coefficient \( f \) is:
\[ f = \frac{V_i^2 - V_e^2 - 30sg}{30s} \quad \text{uphill skid} \quad (3.19) \]

\[ f = \frac{V_i^2 - V_e^2 + 30sg}{30s} \quad \text{downhill skid} \quad (3.20) \]

where

\[ V_i = \text{velocity at beginning of skid, mph} \]
\[ V_e = \text{velocity at end of skid (} V_e = 0, \text{ when vehicle skids to a complete stop)} \]
\[ s = \text{skid distance or length of skidmarks, feet} \]
\[ g = \text{grade of road (} g = 0 \text{ when sliding is on a level road, -ive when sliding is uphill, and +ive when sliding is downhill)} \]
\[ 30 = \text{conversion factor} \]

### 3.3 Skidding on different road surfaces.

All the above formulas are established for one pavement type (that is, involving only one friction coefficient). There are situations in which a vehicle slides or skids on one kind of surface and then ends up on another type of surface and probably on a third type of surface before coming to a stop or hitting something. In such a situation, the frictional force generated on one surface differs from the frictional force generated on
another surface. The motion energy or kinetic energy of the automobile braking on one kind of road surface is given:

\[ KE = \frac{1}{2} \frac{w}{g} V^2 \]  (3.21)

or

\[ KE = Fs \]  (3.22)

where

- \( KE \) = kinetic energy, ft. lbs
- \( w \) = weight of automobile, lbs
- \( g \) = gravitational force acting on automobile, \((32.2 \text{ ft/sec}^2)\).
- \( F \) = frictional force exerted on road by the automobile
- \( s \) = sliding distance of the automobile during braking, feet.

For a two adjacent road surface braking situation, the total motion energy (kinetic energy - \( KE \)) of the vehicle is equal to the frictional force generated on the first surface times the skidding distance on that surface plus the frictional force generated on the second surface times its skidding distance, or
\[ KE = F s_1 + F s_2 \]  \hspace{1cm} (3.23)

where \( s_1 \) and \( s_2 \) represents the skidding distances of the first and second surfaces respectively.

Substituting \( F \) in Equation 2.6 in Equation 3.22,

\[ KE = W (f_1 s_1 + f_2 s_2) \]  \hspace{1cm} (3.24)

where \( f_1 \) and \( f_2 \) represents the friction coefficients of the first and second surfaces respectively.

Equating Equations 3.20 and 3.23 gives,

\[ \frac{1}{2} \frac{W}{g} V^2 = W (f_1 s_1 + f_2 s_2) \]  \hspace{1cm} (3.25)

Canceling out the weight \( W \) on both sides of equation 3.24, Equation 3.24 can be rewritten as:

\[ \frac{V^2}{2g} = (f_1 s_1 + f_2 s_2) \]  \hspace{1cm} (3.26)
Using the conversion factor 1 mph = 1.4667 ft/sec., $g=32.2$, and $V^2 = V_1^2 - V_e^2$, Equation 3.26 becomes:

\[(V_1^2 - V_e^2) \left( \frac{1.4667^2}{2 \times 32.2} \right) = (f_1 s_1 + f_2 s_2) \] (3.27)

Solving equation 3.27 for $V_1$ gives:

\[V_1 = \sqrt{30 (f_1 s_1 + f_2 s_2) + V_e^2} \] (3.28)

This gives the initial speed (in mph) of a vehicle skidding on two adjacent surfaces. The equation is valid for horizontal (no grade) skid only.

If the skidding is on a grade (not horizontal), Equation 3.28 becomes:

\[V_1 = \sqrt{30 (s_1 (f_1 + g) + s_2 (f_2 + g))} \] \textit{downhill skid} (3.29)

\[V_1 = \sqrt{30 (s_1 (f_1 - g) + s_2 (f_2 - g))} \] \textit{uphill skid} (3.30)

where

$V_i$ = initial speed at the start of skidding, mph
\( V_e = \) end speed at the end of skidding, mph, \( V_e = 0 \) if braking brings the vehicle to a complete stop.

\( s_1 = \) skidding distance on first surface, feet

\( s_2 = \) skidding distance on second surface, feet

\( f_1 = \) friction coefficient of first surface (dimensionless)

\( f_2 = \) friction coefficient of second surface (dimensionless)

\( g = \) grade of the surface (it is assumed that the grade on both surface is the same)

\( 30 = \) conversion factor

Similarly, the same reasoning can be applied to skidding on three adjacent surfaces.

\[
V_f = \sqrt{30 \left( f_1 s_1 + f_2 s_2 + f_3 s_3 \right) + V_e^2}
\] (3.31)
3.4 Braking dynamics for Vehicle Pulling Trailers.

The above formulas do not apply to automobiles or trucks pulling trailers in which the trailers are not equipped with brakes. When stopping an automobile-trailer system [5], all the stopping force must be supplied by the towing vehicle. Most of the weight of the trailer is borne by its own wheels and some is carried on the automobile wheels. The frictional force to slow the automobile-trailer is increased because the weight on the braking wheels is increased. The remainder of the trailer weight is carried on its own wheels and hence the available braking force is required to stop a larger mass, and the required stopping distance is increased. In this case, the mass m is the sum of the trailer's weight $W_T$ and the automobile's weight $W_a$ divided by gravity, or

$$m = \frac{W_T + W_a}{g} \quad (3.32)$$

and the braking force is

$$W_a f = \left(\frac{W_T + W_a}{g}\right) a \quad (3.33)$$

Solving for deceleration $a$ in Equation 3.33 gives
Substituting Equation 3.34 into Equation 3.10

\[ a = \frac{fgW_a}{W_T + W_a} \quad (3.34) \]

Using the conversion factor 1 mph = 1.4667 ft/sec., \( g = 32.2 \) and \( V^2 = V_i^2 - V_e^2 \), Equation 3.35 becomes

\[ s = \frac{V^2}{2fg} \left( \frac{W_T + W_a}{W_a} \right) \quad (3.35) \]

Simplifying and rewriting Equation 3.36 gives

\[ s = \left( \frac{V_i^2 - V_e^2}{f} \right) \left( \frac{1.4667^2}{2 \times 32.2} \right) \left( \frac{W_T + W_a}{W_a} \right) \quad (3.36) \]

where

\[ s = \text{stopping (braking) distance, feet} \]

\[ V_i = \text{initial speed, mph} \]
The initial speed, \( V_i \), at the start of skidding of an automobile pulling a brakeless trailer on a level road can be obtained from Equation 3.37 as

\[
V_i = \sqrt{\frac{30fs}{w_a + w_T}} + V_e^2 \tag{3.38}
\]

If the automobile with a brakeless trailer is sliding a grade, the Equation 3.38 becomes

\[
V_i = \sqrt{30s\left[\frac{w_a}{w_a + w_T}\right]f + g} + V_e^2 \quad \text{(downhill skid)} \tag{3.39}
\]

\[
V_i = \sqrt{30s\left[-\frac{w_a}{w_a + w_T}\right]f - g} + V_e^2 \quad \text{(uphill skid)} \tag{3.40}
\]

where

\[
V_i = \text{initial speed at the start of skidding, mph.}
\]
\( V_e = \) final speed at the end of skidmarks, mph.

\( s = \) skid or slid distance, feet.

\( w_a = \) weight of automobile, plus trailer weight carried by the automobile, lbs.

\( w_T = \) weight of trailer carried by trailer wheels, lbs

\( f = \) coefficient of friction of road surface.

\( g = \) grade of the road.

Equations 3.39 and 3.40 may be used for all grades less than 15 per cent.

For grades steeper than 15 per cent,

\[
V_i = \sqrt{30s \cos \phi \left[ \frac{w_a}{w_a + w_T} f + g \right] + V_e^2} \quad \text{downhill} \quad (3.41)
\]

\[
V_i = \sqrt{30s \cos \phi \left[ \frac{w_a}{w_a + w_T} f - g \right] + V_e^2} \quad \text{uphill} \quad (3.42)
\]

where \( \cos \phi \) is simply the cosine of the grade angle. Equations 3.39 and 3.40 are the same as Equations 3.41 and 3.42, because for grade between 15 per cent and 0 per cent, \( \cos \phi \) is between 0.9889 and 1. So it does not make any appreciable difference in Equations 3.41 and 3.42 if the term \( \cos \phi \) is omitted.
From Equations 3.39 and 3.40, the stopping distance for an automobile pulling a brakeless trailer on a grade is

\[ s = \frac{V_i^2 - V_e^2}{30\left[\left(\frac{w_a}{w_a+w_T}\right)f-g\right]} \quad \text{downhill skid} \]  \hspace{1cm} (3.43)

\[ s = \frac{V_i^2 - V_e^2}{30\left[\left(\frac{w_a}{w_a+w_T}\right)f+g\right]} \quad \text{uphill skid} \]  \hspace{1cm} (3.44)
4. ACCIDENT RECONSTRUCTION AND SKID PREVENTION

4.1 ACCIDENT RECONSTRUCTION PROGRAMS

An overview of selected computer programs for vehicle accident reconstruction is addressed in this chapter. The last part of the chapter takes a look at some of the counter-measures against automobile skidding. Some of the accident reconstruction programs have become well established in the last 10 years whereas others are new. They provide simulations of collision and vehicle trajectory to varying levels of complexity, sophistication, and ease of use. Some of these programs are as follows:

4.1.1 CRASH—Calspan Reconstruction of Accident Speeds on the Highway.

CRASH program is designed to compute the impact speeds of two colliding vehicles. It is designed for direct impact-type collision involving two vehicles. It has undergone series of updating to CRASH2 and CRASH3. CRASH has a "damage only" analysis option which is currently used in the National Accident Sampling Systems (NASS) accident data collection program [16]. About 45% of the accidents investigated in the National Crash Severity Study (NCSS) are measured by CRASH.

The major advantage of using CRASH is that it is completely independent of traditional reconstruction methods that use skid distances and momentum. The CRASH algorithm, which is based on the method proposed by Campbell [17], requires comparative crash test data and crush measurements taken from the accident vehicles or
estimate from photographs. Since it is often difficult if not impossible to adequately estimate drag factors, especially when the automobile traverses multiple surfaces with different friction coefficients, the damage method of reconstruction, as used in CRASH3 program is gaining more importance than traditional reconstruction methods. The central disadvantage of the damage analysis of CRASH is that it can only yield information about speed change or the relative approach speed of two colliding vehicles. Road speed or speedometer speed cannot be obtained by the damage method alone. In addition to the damage method for reconstruction, CRASH3 calculates post-impact velocities based on the distance between the point of impact and the point of rest, surface friction, average rolling resistance of the tires, direction of rotation, number of revolutions, and the curvature of the center-of-gravity path.

The source code Fortran listing for CRASH3 is available on tape from the National Center for Statistics and Analysis (NCSA), National Highway Traffic Safety Administration (NHTSA), U.S. Department of Transportation (DOT). Operational versions may be run via modem at 1200 baud using the computer facilities at the University of Michigan Transportation Institute, or at the Boeing Computer Services Company, Vienna, Virginia. The CRASH3 program has also been ported to personal computers (PCs) with runtime offered for sale for several popular PCs [18].

4.1.2 EES-ARM - Equivalent Energy Speed Accident Reconstruction

The EES-ARM program has been widely used in Europe for speed reconstruction
in automobile crashes. It is designed to evaluate the collision phase relationships, using physical principles and approximations customarily used in hand calculations [19, 20]. Like the crush damage calculations in CRASH, EES-ARM automates the methods normally usable for hand calculation of collision-phase speed based on the principles of momentum and energy. Common to CRASH3, it requires that the user provide quality input regarding the angular relationships in the collision. The energy inputs are required in the form of an "energy equivalent speed" (EES) for each car, based on interpretation of crash test data and adjustments for test vehicle mass variations. Regression equations developed by Zeidler are used for the EES values. The program is reported to be available through DEKRA in Stuttgart [21].

4.1.3 HVOSM-- Highway Object Simulation Model

The first chronology of the U.S. Government-sponsored reconstruction program was the HVOSM developed under Federal Highway Administration Contract CPR -11-3988 between 1966 and 1971. This model is intended to describe the three-dimensional motion of an automobile in space, including interaction with roadway, shoulders, ramps, berms, etc. It is supplied in two versions: HVOSM-RD which emphasizes road or highway design and HVOSM-VD which emphasizes vehicle dynamics. The HVOSM-RD makes provisions for simplified modeling of collisions with fixed objects and the HVOSM-VD is for the evaluation of vehicle trajectory due to launch, vault, or handling maneuvers. This last version contains built-in models for
engine torque and drag, hydraulic brake pressure versus brake torque at a given wheel and also allows for assumptions about driver control inputs. The program is available for use, either through DOT contract computer auspices, or it may be obtained on tape from the FHWA.

4.1.4. IMPAC—Impact Momentum of a Planar Angled Collision

The IMPAC program [22] is intended to provide a straightforward and simple analysis of angled collisions, providing something that allows the user to at once avoid the tedious hand calculations of momentum and the complexity of SMAC [23] while achieving a useful technical result. IMPAC is posed as an initial value problem rather than as a final value problem. The user specifies pre-impact conditions and the program calculates post-impact conditions. The IMPAC program provides a simple, easily used collision model to reconstruct accidents that are beyond hand calculation. It has also been used in combination with vehicle trajectory simulation (VTS) for trajectory analysis, as a preprocessor for SMAC to reduce the number of runs required to obtain a reconstruction, and to study sensitivities in proposed crash test alignments, both car-to-car and car-to-barrier. IMPAC may be accessed at CSE via telephone or modem at 1200 or 2400 baud from a variety of terminals or personal computers. The program operates on an HP-9000 series 500 computer under the UNIX operating system (HP-UX). Those who wish to experiment with the IMPAC or VTS programs may do so by contacting the authors to establish a dial-up connection.
4.1.5 SMAC—Simulation Model of Automobile Collision

The SMAC computer program was developed by McHenry for the U.S. Department of Transportation (DOT) [23, 24]. It simulates the collision phase and post collision phase based upon the speeds, positions and orientation of the vehicles. To conduct SMAC runs, vehicle geometry, mass, yaw inertia, and tire properties, together with time-dependent braking and steering, data are tabulated in program input for each vehicle, inter-vehicle friction and restitution are selected, and one or two roadway friction regions are described. Initial conditions of velocity and position in two rectangular and one angular coordinate are described for each automobile, and simulation control inputs are inserted to initiate and terminate the computer run. The computer calculates individual tire forces and inter-vehicle crush forces for this initial-value problem at preassigned time steps by a Runge-Kutta integration scheme. SMAC was revised later to a new model called PRED [25, 26] due to some programming errors and conceptual problems in the early phase of its development. The source code Fortran listing for SMAC is available on tape from the National Center for Statistics and Analysis, NHTSA, U.S. Department of Transportation. Operational versions may be run via modem at 1200 baud using the computer facilities at the University of Michigan Transportation Institute or at the Boeing Computer Services Company, Vienna, Virginia.

4.1.6 TBS—Tractor Braking and Steering simulation

The TBS simulation was developed at the Highway Safety Research Institute
(HSRI) under the sponsorship of the Motor Vehicle Manufacturers Association (MVMA) in addition to earlier more comprehensive programs [27, 28]. This simulation contains a simplified vehicle model for predicting the directional response of commercial vehicles to braking or steering inputs, or both. The simulation consists of two interactive computer programs—one for a straight truck (TBSTR) and the other for a tractor-trailer (TBSTT).

The mathematical model for TBS was constructed using the model developed by Leucht [29] as a starting point. The TBS simulation was formulated and programmed to describe the directional dynamics of a tractor-trailer. The program is designed so that the user answers questions or enters data in response to questions or commands from the computer. In addition, data for the program may be optionally input from a file. The source code Fortran listing for TBS is available on tape from (UMTRI) [30]. Operational versions of TBSTT and TBSTR may be run via modem at 1200 baud using the computer facility at UMTRI.

4.1.7 VTS—Vehicle Trajectory Simulation

The VTS program [31] simulates the trajectory of one vehicle on a horizontal surface. Its application to accident reconstruction is to study the preimpact and post-impact motion of a vehicle (automobile or two-axle truck, no trailers). Preimpact motion is studied to define vehicle capabilities in time, such as steering and braking vehicle responses based on assumed driver inputs. VTS's use in simulation of post-impact motions helps to define
post-impact velocities and rotation rates or to determine the average deceleration of a damaged rotating automobile. These parameters are needed for subsequent collision calculations. Given an estimate of the point of impact and point of rest, VTS is used in an iterative manner to estimate launch conditions that produce plausible trajectories to rest.

4.2. Skid Prevention and Counter-Measures for Skid Reduction

Loenard [32] suggests that skid prevention is not the prevention of "sliding" and "slipping" motions, per se, from occurring on the roadway, but rather the prevention of accidents in which drivers cannot exercise control over their position in space and time. He further suggests that "Skid prevention" must realistically be interpreted as the reduction of the probability that drivers will lose control over their path, particularly when roadways with degraded frictional qualities must be negotiated. Steer moment, brake torque and drive torque created by the driver are factors which bear on the ease with which the driver can control his vehicle, namely, the influence and the ability of drivers to modulate their control inputs and thereby generate forces for maneuvering appropriate to the circumstances that prevail. Knowledge thus exists with respect to how the indicated factors influence the ability of drivers to minimize the occurrence of wheel spin or wheel lock [32]. These factors should be integrated into the design process. Attention should be given to the ergonomic implications of these design variables in increasing or reducing the potential for loss of control. Loenard [32], further emphasizes
that design factors influencing the redistribution of loads among the wheels of a maneuvering motor vehicle are highly constrained and that instead of opportunities to optimize this redistribution for minimizing the loss of control potential, it appears that the growing pressure for smaller, more efficient cars will lead to a vehicle population with shorter wheelbase and lesser track widths. This trend will, in turn, mean increased maneuvering induced loads and reduced maneuvering limits on dry and perhaps on wet roads as well. They are basically three ways of preventing loss of control [33].

1. By appropriate design of the brake distribution.

2. By adopting a special method of applying the brakes.

3. By use of an anti-locking device to release the brakes on all or some of the wheels when locking is imminent.

4.2.1. Brake distribution

The most direct method of ensuring stability during braking is to design the brake distribution so that (for surfaces having the highest resistance to skidding, and for a lightly loaded vehicle) it matches the weight distribution, taking into account load transfer during braking. Brake distributions is either accomplished by designing it a fixed ratio (e.g. more braking power for front wheels than rear wheels) or variable ratio (i.e. variation of brake distribution to changes in weight distribution) [33].
4.2.2. "Pumping" the brakes

This special technique is to momentarily release brakes and reapply as rapidly as possible. Tests carried out to compare the pumping technique are described in [34]. It was found that repeated rapid application and release of the brakes enabled the driver to keep a car on a straight path on a variety of surfaces, both wet and dry. Furthermore, the braking distances achieved by using the pumping technique are generally slightly shorter than those achieved by braking with locked wheels. The rate of pumping was in the order of two to three applications per second.

4.2.3. Antilock brake systems

The purpose of antilock brakes is to take the fullest advantage of available tire-pavement friction capabilities without locking the wheels and losing vehicle control. Antilock brake systems are designed to achieve and maintain the peak coefficient of tire-pavement friction which maximizes the braking effect. In applying the brakes of a car so as to lock the wheels, the following sequence of events takes place as described by graph of Figure 4.1. At the instant when pressure is applied to the brake pedal, the wheels are rotating freely, the slip (the percent decrease in the angular speed of a wheel relative to the pavement surface as a vehicle brakes) is zero, and of course the braking force is zero too.
Figure 4.1 Variation of Braking Friction Coefficient with per cent slip
As the pressure is increased, the slip increases until at some critical value the braking force reaches a maximum; if the slip increases still further, the braking force decreases until finally the wheel locks, at which time the slip is 100%. Applying antilock device to rear wheels only prevents the rear wheels from locking and large uncontrollable angular deviations during emergency braking can be avoided. Antilock brake systems operate by monitoring each wheel for impending lockup. When wheel lockup occurs or is anticipated, the system releases brake pressure on the wheel. When the wheel begins to roll freely again, the system reapplies braking pressure. The system constantly monitors each wheel and readjusts the brake pressure until the wheel torque is no longer sufficient to lock the wheel. Present antilock brake systems are controlled by onboard microprocessors.

An NHTSA study [34] of the performance of a commercially available antilock brake system on a two-axle, single-unit found a 15 percent reduction in the braking distance for a straight line stop from 60 mph on a wet, polished concrete pavement surface with an SN40 of approximately 30 (similar to the surface used by the AASHTO Green Book in specification of stopping sight distance standards). Tests on other pavement surfaces and in other types of maneuvers found decreases in braking distance up to 42 percent with antilock brake system [35]. Furthermore, in addition to improving the braking efficiency by operating closer to the peak braking friction coefficient, antilock brake systems should also minimize the increase in braking distance caused by driver inexperience.
CHAPTER 5

5. GUIDELINES FOR SOFTWARE DESIGN

5.1. INTERACTIVE PROGRAMMING

Any program that requires user input is by definition, an interactive program. This is accomplished through series of response levels or response sequences. The simplest kind of response is a key press that indicates acknowledgment. The next level of response is a choice from a list of options in a menu or a multiple choice question. In this software package, the user chooses a particular response by typing a single letter or number.

In designing an interactive system, a thorough task analysis should be conducted to provide information for a proper functional design. Design alternatives should be evaluated based on rapid task performance, low error rates, higher user satisfaction and ease of retention. Also the design should begin with an understanding of the intended users. There is a category of users who do have any knowledge of computer issues, another category have knowledge about computers but do not know the different function keys or commands used to instruct a computer to perform a certain task, and a third category of users who have knowledge about computers and how to use them and just require rapid response and brief feedback. The software is designed for a wide variety of user population. It's user's guide simple and straight forward step by step procedure makes it easily accessible to any category of users who can start up a computer and read simple instructions.
Some of the characteristics of a good interactive program outlined in [47], includes:

1. A program is valid if it is accurate, complete and current.
2. A program that has high usability will take very little time to learn and result in few user errors.
3. A good interactive software must capture attention and facilitate performance. This means that screen displays must be stimulating and well organized.
4. The program must offer a lot of flexibility over pace and sequencing. It is imperative that the user have ample control of the program.
5. The program must be consistent in the way it works and be free from bugs. This means that the program must be very reliable in order to be trusted and believed.
6. Finally, the program must be perceived as useful by users, that is, they must feel that they have accomplished something by using it.

Like all programs, problems are bound to be encountered right from the initial stage till the end if proper tasks analysis and design guidelines are not completely spelt out. To aid the designer in setting guidelines, he/she can look at typical problems normally associated with a software and use these problems as the basis for design guidelines.

Some of the typical problems in a software outlined in [47], includes:

1. Rigid- Too much precision required does not allow variability in response format and for that matter does not allow errors to be corrected. It is therefore not a good idea to expect the user to respond precisely.
2. Inconsistency is a frequent source of user frustration. It translates into a perception of unpredictability by the user.

3. Too much information at one time on displays often overload users. Human performance limitations (particularly memory capacity) need to be taken into account in program design.

4. Sometimes the terminology used in for example, command error messages, directions, etc. often makes simple things very complicated for users.

5. Providing feedback helps users keep track of where they are in the program.

6. Poor error handling and recovery makes programs fragile and a source of annoyance to users.

5.2. SOFTWARE SUPPORT

Even a well designed program may fail if it is poorly supported. Some form of user support is therefore needed for almost all software. A user's guide manual is put together as a support for the software package, explaining step by step how to use the software and what to expect at each level of command execution. This manual is available in Appendix F. The manual is especially useful for procedures which are not repeated often enough to warrant the user learning the procedures. It also combines both installation and application guide with specific procedures and instructions explaining how to use the software.
5.3 DISPLAYS

Display screens are often the primary factor used to judge the quality of programs. The screen displays in the software are intended to capture attention and facilitate performance by its well organized and colorful menus. The different types of screens in this software are:

1. Data entry in which users enter some data to such parameters as speed, distance, time, coefficient of friction, etc..

2. Informational screens presents information requested by the user, or general information pertaining to function or parameter the user is interested in finding.

3. Calculation and results screen allows the users to view the results of any computations in the program.

4. Menu screens allows the user to select options in a program menu.

5. Question/Answer screens allows the user to answer questions leading to the computation of whatever the user is attempting to compute.
CHAPTER 6

6. SOFTWARE SPECIFICATIONS-- Assumptions and Requirements

6.1. Assumptions for the model.

One of the key assumptions made in developing the model for the software is that there is no variability in the system and that the output is uniquely determined by input data/information and established rules. Some of the rules are as follows.

1. Brakes must be capable of locking the wheels and bring the vehicle to a complete stop.

2. The stopping vehicle does not collide into or with something that is capable of changing or altering its speed and trajectory. Exceptions includes automobile pedestrian collisions, collision into a fence post or wooden road sign.

3. Skidding distance only assumes a straight line.

4. Weight is not a factor in the stopping distance.

5. Coefficient of friction of the road surface on which skidding takes place remains constant throughout the skidding period.

6. Energy dissipated by each of the four tires of the automobile remains the same.

7. When all tires are sliding, rolling resistance, engine braking and the amount of wheel braking are irrelevant or insignificant. They all add up to enough to lock the wheels.

8. Minimum slowing is when only the rolling resistance and air resistance are the only forces retarding the vehicle.
9. Maximum slowing is when brakes have locked all wheels and the vehicle simply skids to a stop. The drag factor is then the coefficient of friction between tires and pavement during the skid.

10. Braking requirements shall be applicable under all conditions.

11. Type of vehicle has little effect on the distance required to slid to a stop.

12. Tread pattern of tire is assumed to be insignificant to alter the slid to stop distance.

13. Road temperature is also insignificant.

14. Braking effort is designed to be equal on the two ends of each axle.

15. Vehicle weight is uniformly distributed between the two wheels of each axle, i.e. the right and left wheels are equally loaded.

16. The skid distance is taken to be the distance moved by the vehicle's center of mass.

6.2. REQUIREMENTS FOR THE MODEL

6.2.1. User requirements:

To design a good program, it is absolutely essential to know what the user wants (and does not want). Defining user requirements is a major focus of human factors engineering. Some of the design goals of human factors in terms of user requirements as outlined in [47], are:
1. **Time to learn** - How long does it take for a typical member of the target community to learn how to use the commands relevant to a set of tasks. Obviously, the more time it takes the user to learn certain commands before executing a particular task, the more frustrated the user will become and the less likely that the program will get any patronage.

2. **Speed of performance** - How long it takes to carry out the benchmark set of tasks? The longer it takes, the less likely the various types of users will use it, especially experience users who among other things are conscious about speed.

3. **Rate of errors by users** - How many and what kinds of errors are made in carrying out the benchmarks of set of tasks. Error rate must be as low as possible and the errors must be such that they can be easily understood and corrected.

4. **Subjective satisfaction** - How much did user like using aspects of the system?

5. **Retention over time** - How well do users maintain their knowledge after an hour, a day or a week?

Taking the above into account, the user requirements in this software package are: 1) that the user knows basic vehicle dynamics and the natural laws of motion, 2) that the user knows how to start a computer, 3) how to use the disk drive, understand and be able to execute commands on the computer, and 4) he/she is not required to have any computer programming knowledge or experience. All he/she has to do is insert a computer diskette (3.5", 5.25"), depending on the type of computer he/she is using, turn on the machine and
access the program on the diskette. When the program is stored on a hard drive, it is even easier, the user only needs to turn on the machine and access the program from the hard drive.

6.2.2. Hardware Requirements.

The software is developed on a 386 SX IBM compatible machine with two floppy drives (3.5" with 1.44 mega bytes and 5.25" with 1.2 mega bytes) and a hard drive of 50 mega bytes. However, it can run on the IBM PC series or its compatibles including PC 286, PC 386, PC 486 with at least one floppy disk drive and 640 Kilo bytes of RAM. It is implemented in Turbo C for the MS-DOS version 5.0 and will run on newer MS-DOS versions. Also needed is a graphic adapter (either EGA - Enhanced Graphics Adapter or VGA - Video Graphics Array) with which only the "EGAVGA.BGI" file is needed. A math co-processor may be added to speed up the calculations but not a requirement. In order to obtain a hard copy of the final results an on-line dot matrix printer will be needed. The executive code is generated on the basis of the 80186 / 80286 CPM, so that the computer CPU must be at least Intel 80286.
6.2.3. Software Requirements.

1. Given that a vehicle leaves skidmarks on a level road with say, the front wheels. Given also the road friction coefficient and the final speed of the vehicle, the software will be able to generate the following output:
   
   a. the initial speed, skidding time and rate of deceleration
   b. plot the graph of speed as a function of distance
   c. plot the graph of speed as a function of time
   d. plot the time as a function of distance
   e. show tables of the above plots.

2. Given the initial speed, skidmarks length and coefficient of friction, the package will generate the following output:
   
   a. the final speed, skidding time, and rate of deceleration
   b. plot the graph of speed as a function of time
   c. plot the graph of speed as a function of distance
   d. plot the distance as a function of time
   e. show tables of the above plots

3. Assuming the coefficient of friction of the road is not known, but the speed and distance are known, the software will determine:
   
   a. the friction coefficient, skidding time and rate of deceleration
   b. plot the graph of the time as a function of speed
c. plot the graph of the distance as a function of speed  
d. plot the graph of the speed as a function of time  
e. show tables of the above plots  

4. The package also calculates the time from beginning to end of skidding and plots the time profile.  
5. Also in situations where the skidding vehicle is pulling a brakeless trailer, the package can be used to handle the above cases.  
6. The package calculates the driver perception-reaction distance and total stopping distance given the driver perception-reaction time.  
7. The package can also be used as a calculator. That is to say, by entering the values of any two parameters from a set of four parameters (speed, distance, time and friction coefficient), the software automatically generates values for the other two parameters without further questions as in the other options.

When all the wheels are locked during skidding, the total drag is equal to the coefficient of friction of the road (only when road is level). If some wheels are sliding and some are rolling, the software first calculates the total drag factor. Also the software will perform the above tasks for skidding on grades (uphill and downhill) and skidding on more than one surface type (maximum of three adjacent surfaces).
6.2.4. Design Requirements.

1. Overall goal is user friendliness

2. Menu-driven

3. User input
   a. objective input (‘Y’ or ‘y’ for YES response, ’N’ or ’n’ for NO response).
      Letter alphabet or number entry for options (all error-free entry). Requires 
      repeated trials if wrong entry is made till the right entry is made.
   b. subjective input- Requires realistic input data. Unreasonable inputs, e.g.
      negative value entry for speed, alphabet or any character entry are 
      promptly rejected with a message to try the entry again until the right entry 
      is made.

4. Screen displays are designed to lead the user to successfully run the program and 
   get the desired results.

The program must be well documented to make it easily understandable. In this regards, 
every function (subroutine) in the program has a description at the beginning explaining 
what that function or subroutine is all about and what is needed. Also every variable 
name or sub-name of a structure variable is explained or described in Appendix C. 
Descriptive comments are also to be found within the program codes or on separate 
comment lines. User output results can be verified from graphs and tables available in 
Appendix B. The user can also plot the graphs using the software. The graphs consists
of all possible relationships between the parameters dealt with in the package. Also tables are provided for various grades (0, -0.03, 0.03, -0.06, 0.06) on the general relationship between speed, distance, time, and friction coefficient, so that if the user is using any of these grades he/she can verify his/her results from those tables available in Appendix B.
CHAPTER 7
STRUCTURE OF THE SOFTWARE PACKAGE

The Software is designed to be fully menu-driven. The main menu is composed of five major function prototype options plus the 'Quit' option. The five options are: Speed (Initial or final), Skid_Distance, Skid_Time, Coeff_Friction, Calculator major options. Figure 7.1 shows the main menu pop-up structure and Figure 7.2 gives the flowchart of the skeleton structure of the software.

MAIN MENU

1. SPEED
2. SKID-DISTANCE
3. SKID-TIME
4. COEFF-FRICTION
5. CALCULATOR
6. QUIT

Figure 7.1 Pop-up window of the Main Menu
Figure 7.2  Flowchart of the Software skeleton
7.1 'SPEED' Function Prototype Option

The first function /option in the main menu is 'SPEED'. The SPEED menu is shown in flowchart structure in Figure 7.3. Figure 7.4a shows the sub-menus which also represents options: Initial Speed, and Final Speed and an option to 'QUIT'. The user at this stage can chose what type of speed he/she wants to find. Each of these sub-menus will take the user to yet two more menu types denoted as "Road surface' and 'Grade of road'. The 'Road surface' menu consists of three options namely: One surface, two surfaces and three surfaces. These options allows the user the flexibility to run the software for skidding involving one road surface, two adjacent surfaces, or three adjacent surfaces. The 'Road Structure' sub-menu consists of three options namely: Level, Uphill or Downhill. After determining the road surface, the user in this 'Grade of road' option has to chose whether the skidding is on a level, uphill or downhill road. Figures 7.4b and 7.4c shows the 'Road surface' and 'Grade of road' sub-menus respectively. Figures 7.5 and 7.6 shows the flowcharts for the initial speed and final speed options respectively.
Figure 7.3  Flowchart of the SPEED option menu
### SPEED
- A. Initial-Speed
- B. Final-Speed
- C. Main Menu
- D. Quit

### ROAD SURFACE
- A. One Surface
- B. Two Surfaces
- C. Three Surfaces
- D. Return to Speed Menu

### GRADE OF ROAD
- A. Level
- B. Uphill
- C. Downhill
- D. RETURN TO ROAD SURFACE

Figure 7.4  Popup windows of sub-menus
Figure 7.5 Initial Speed Function flowchart.
Figure 7.6 Final Speed Function flowchart.
When the skidding is on the same type of road surface and the surface is level, the drag factor produced by the wheels of the sliding vehicle will be equal to the friction coefficient of the road if all the wheels are locked during the skidding. If some wheels are rolling instead of sliding, the wheels will produce different drag factors. The total drag factor has to be calculated by a function prototype called 'Drag Factor1'.

The situation is much complicated when skidding involves more than one type of road surface. In such a situation, even if all the wheels are locked during skidding, the drag factor on the front and rear axles will be different because the wheels are skidding on different road surfaces with different coefficient of frictions. This situation is tackled under the function prototype 'Drag factor2' in the software package and discussed in chapter 2. However, if all the wheels produce skidmarks on one surface and then traverses into another surface with all the wheels leaving skidmarks on it, then the drag factor produced by the wheels will be the same on each surface and hence there would not be the need to calculate the total drag factor. Figure 7.7 demonstrates this point. In this figure, a complete skid is made by all four wheels on an asphalt surface with the rear wheels skidmarks superimposed on the skidmarks left by front wheels for a distance of $s_1$ units and traverses into gravel for a complete skid of $s_2$ units. At this stage, if the surfaces are level, the software will take the friction coefficient $f_1$ of the asphalt surface as the drag factor during the $s_1$ units skidding and the friction coefficient $f_2$ of the gravel surface as the drag factor during the $s_2$ units skidding and use them to compute the initial speed and other parameters the user may be interested in finding.
As discussed in chapter 2, if some wheels are sliding on one type of road surface while other wheels slides on another surface, the total drag factor of has to be calculated. Figure 7.8 shows examples of situations under which the software will first calculate the front and rear axle drag factors and if the two drags are not equal will proceed to calculate the total drag factor.

In Figure 7.8a, one skidmark is on the shoulder and the other three skidmarks are on the asphalt surface. Using $F_d$ and $R_d$ to represent the front and rear axles drags respectively, and $f_1$ and $f_2$ as the friction coefficients for the asphalt and shoulder respectively, the software calculates the axles drag factors. Thus

$$F_d = \frac{(f_1 + f_2)}{2}$$

$$R_d = f_2$$

Similarly, in Figure 7.8b,

$$F_d = f_1$$

$$R_d = \frac{(f_1 + f_2)}{2}$$

In Figure 7.8c,

$$F_d = f_1$$

$$R_d = f_2$$

and in Figure 7.8d,

$$F_d = \frac{(f_1 + f_2)}{2}$$

$$R_d = \frac{(f_1 + f_2)}{2}$$
Figure 7.8 Examples designed to show how the front and rear axles drag factors can be different in a complete skidding of a four-wheel vehicle on two adjacent surfaces.
As mentioned above, if $F_d$ is not equal to $R_d$, then the total drag factor is calculated under the function prototype $\text{Combined\_Drag}$.  

Figure 7.9 shows a vehicle making a complete skid on three surfaces. The same concept as in Figure 7.7 is used to compute the speed, skidding time and deceleration.
Figure 7.9 Complete skidmarks from a four-wheel vehicle sliding on asphalt surface for $S_1$ distance and then traverses into a gravel shoulder for $S_2$ distance, and finally sliding $S_3$ distance in dirt before coming to a stop. The rear wheels skidmarks are superimposed on the front wheels skidmarks. $f_1$, $f_2$, and $f_3$ are the friction coefficients of the asphalt, gravel and dirt respectively.
7.2 'SKID_DISTANCE' Function Prototype option

This option is designed to enable the user to determine theoretically the distance a vehicle skids for a given initial speed and a given drag factor, without practically measuring the skidmarks (i.e. if skidmarks are made by the sliding vehicle). Like the INITIAL_SPEED option, the skidding distance is also affected by the type of drag produced by the vehicle and the elevation of the road surface on which the skidding takes place.

To compute the skidding distance under this option, the user must provide the values to any of the following parameter sets:

- initial speed, final speed, coefficient of friction, grade of the road.
- initial speed, final speed, skidding time.
- skidding time, coefficient of friction, grade of the road.

This option also plots the skidding distance profile and tables corresponding to each plot.

Figure 7.10 shows the flowchart for the Skid_Distance function.
Figure 7.10 Skidding Distance Function flowchart.
7.3 'SKID_TIME' Function Prototype option

The SKID_TIME option is primarily to compute the time during which the vehicle produced the skidmarks. This is accomplished when the user inputs the values of one of the set of parameters shown below.

- initial speed, final speed, friction coefficient, grade of the road
- initial speed, final speed, skidmarks length
- skidmarks length, friction coefficient, grade of the road.

Also available under this option are plots of the time profile and their corresponding tables. Figure 7.11 shows the flowchart of the skid_Time function.
Figure 7.11 Skidding Time Function flowchart.
7.4 'COEFF_FRICTION' Function Prototype option

This option is primarily used to compute or estimate the coefficient of friction of the surface on which skidding or sliding takes place. For a level surface, the coefficient of friction is the force $F$ required to slide a vehicle or object divided by the weight $W$ of that vehicle or object. Since the force $F$ cannot be easily measured or given as a quantity, the user has to provide the values for any of the parameter sets listed below.

- initial speed, final speed, skidding distance, grade of the road
- initial speed, final speed, skidding time, grade of the road
- skidding distance, skidding time, grade of the road.

The option also plots the coefficient of friction profile and the corresponding tables. Figure 7.12 shows a flowchart of the Coeff_Friction function.
Figure 7.12 Friction Coefficient Function flowchart.
7.5 'CALCULATOR' Function Prototype option

The CALCULATOR option is designed to work like a nomograph (arrangement of scales) which shows the relationship between sliding distance, speed, time and drag factor. In this option, if any two of the above four quantities are given by the user, the third and fourth quantities can be easily obtained. In other words, it works like a calculator, in that if the user provides for example, the values of the sliding speed and distance, he/she automatically gets the values of the skid time and drag factor. Figure 7.13 shows a flowchart for the Calculator function.

The CALCULATOR option is demonstrated graphically by the Traffic Template nomograph on Figure 7.14. The figure shows the scales for the speed, distance, time and drag factor placed side by side. Given any two of them, e.g. speed and distance, a straight line is drawn through the points indicating the given values of the speed and distance. If that straight line is extended to cut the time scale and drag factor scale, the point at which it cuts the latter scales gives the values of the time and drag factor corresponding to the speed and distance given.
Figure 7.13 Calculator Option flowchart.

- **V**: Speed
- **d**: Distance
- **t**: Time
- **f**: Drag factor
Figure 7.14 Traffic Template Nomograph relating stopping distance, time, initial speed and average drag factor.
7.6 Verification and Validation of the Software.

In this section, a number of example problems have been formulated for analytical computation of the parameters (initial/final speed, skidding distance, skidding time and coefficient of friction) dealt with in the developed software with the view of comparing the computed results with those generated by the software. Such comparison will thus enable us to affirm the validity of the software. Also a further step is taken to verify the accuracy of the results generated by the software by comparing them with the solutions to those same problems, solved by other means or methods.

7.6.1 Example 1: Measurement of the coefficient of friction.

An automobile skidded a distance of 50 meters (164 feet) from an initial speed of 70 km/h (43.5 mph) before coming to a stop. What is the coefficient of friction of the road surface on which the skidding occurred. Assume that all the automobile wheels were locked during the sliding and the road was level.

\[ f = \frac{V_i^2 - V_e^2}{30s} \pm g \]

\[
\begin{align*}
  f & = \text{coefficient of friction of the road (dimensionless)} \\
  V_i & = \text{initial speed of automobile, mph} \\
  V_e & = \text{final speed of the automobile, mph} \\
  s & = \text{skidding distance, feet} \\
  g & = \text{grade}
\end{align*}
\]
Software requested input to solve the above example:

- initial speed = 44 mph
- final speed = 0 mph
- braking distance = 164 feet
- grade of surface = 0.0

Output of software solution:

- friction coefficient = 0.39
- skidding time = 5.1 seconds
- deceleration = 12.7 ft/sec²

The coefficient of friction comes out exactly the same, i.e. \( f = 0.39 \) as the analytical solution. These results can verified to be exactly the same with Exhibits 3-1 and 3-2 from J. Stannard Baker book on Estimating Stopping distance and Time for Motor Vehicles [5].
7.6.2 Example 2: Calculating the initial speed just before wheels lock.

An automobile skidded for a distance of 30 meters (98.4 feet) on a dry asphalt road \((f=0.7)\) before coming to a complete stop. Assuming that all the automobile wheels were all locked and sliding and that the road surface was level, at what initial speed was the automobile travelling just before the wheels locked.

\[
V_i = \sqrt{30s(f \pm g)} + V_e^2
\]

- \(V_i\) = initial speed of the automobile, mph
- \(V_e\) = final speed of the automobile, mph
- \(s\) = skidding distance, feet
- \(f\) = coefficient of friction (dimensionless)
- \(g\) = grade of the road

\[
V_i = \sqrt{30 \times 98.4 (0.7 \pm 0)}
\]

\[
V_i = 45.5 \text{ mph}
\]

Software requested input to solve the above example:

final speed \(= 0\) mph
braking distance = 98.4 feet
grade of surface = 0.0
friction coefficient = 0.7

Output of software solution:
initial speed = 45.5 mph
skidding time = 2.9 seconds
deceleration = 22.5 ft/sec^2

The result of the initial speed is exactly the same as the analytical solution. This is also verified to be the same with the result from Exhibits 3.1 and 3.2 from J. Stannard Baker book on Estimating Stopping Distances and Time for Motor Vehicles [5].

7.6.3. Example 3: Calculating the Stopping (braking) distance.
A vehicle traveling on a dry, clean portland concrete pavement having an average coefficient of friction f=0.7 suddenly started skidding before coming to a complete stop. The initial speed of the vehicle was 42 km/h (26 mph). Assuming the vehicle was on a level concrete road and all its wheels were sliding, what is the minimum stopping distance (distance travelled from the beginning of skidding to the end of skidding) of the vehicle?
s = stopping (braking) distance, feet
f = coefficient of friction of road (dimensionless)
$V_i$ = initial speed, mph
$V_e$ = final speed, mph
g = grade

\[ s = \frac{V_i^2 - V_e^2}{30(f \pm g)} \]

\[ s = \frac{26^2 - 0^2}{30(0.7 \pm 0)} \]

\[ s = 32.2 \text{ feet} \]

Software requested input to solve the above example:

initial speed = 26 mph
final speed = 0 mph
grade of surface = 0.0
friction coefficient = 0.7

Output of software solution:

skidding distance = 32.2 feet
skidding time = 1.7 seconds
deceleration = 22.5 ft/sec^2
The results can be verified to be exactly the same as those in Exhibits 3.1 and 3.2 from J. Stannard Baker’s book on Estimating Stopping Distance and Time or Motor Vehicles [5] gives the same results.

7.6.4 Example 4: Calculating time taken to produce the skidmarks.

A vehicle skidded a distance of 20 meters (65.6 feet) on a smooth and level icy road before coming to rest. Assuming the coefficient of friction of the road was 0.20, and all its wheels were sliding, what time did it take to skid to a stop?

\[ t = 0.452 \sqrt{sf} \]

where

\[ t = \text{skidding time, seconds} \]
\[ s = \text{skidding distance, feet} \]
\[ f = \text{coefficient of friction (dimensioless)} \]

\[ t = 0.249 \sqrt{65.6/0.20} \]

\[ t = 4.510 \text{ seconds} \]
Software requested input to solve the above example:

- skidding distance = 65.6 feet
- final speed = 0 mph
- grade of surface = 0.0
- friction coefficient = 0.2

Output of software solution:

- skidding time = 4.5 seconds
- initial speed = 19.8 mph
- deceleration = 6.4 ft/sec²

This analytical solution and the solution generated by the software is exactly the same. Also Exhibits 3.1 and 3.2 from J. Stannard Baker's book on Estimating Stopping Distance and Time for Motor Vehicles [5] gives the same result.

The above four examples are ideal problems all based on complete skidding horizontal road surfaces. However, the software also covers grade (downhill and uphill) skidding, uneven weight distribution on the wheels in which some wheels may not be locked during braking and the possibility of an automobile carrying a brakeless trailer at the time of skidding. The same concept is extended to skidding on two and three surfaces.
7.7 User Test and Evaluation of Software Package

The problems below are selected from books for testing and evaluating the software. The problems are solved manually and then solved using the software. The results are then compared with the solutions from the books where those problems are taken. Also a small group of users are presented with the problems and asked to solve them manually and then through the software. They are then given the questionnaire forms to get their comments. The completed forms are available in appendix B. Table 7.1 shows the summary of the comments the users made after they used the software. The users response are generally satisfactory. Most of the users acknowledged they would not have been able to derive the formulas needed to solve the problems, and with the software, one does not need to derive any formula or do any calculations. What is not included in this summary are some initial problems some of the users identified. These problems were: 1) mixed units (English units mixed with metric units at some places in the software; 2) not enough information to solve some of the problems presented to them; and 3) questions not very clear enough. These problems are not recorded in the summary because the software was revised to take care of all them and the final version was presented to the same users. So the summary only reflects the final version of the software.
**TABLE 7.1**

Summary of feedback from users who conducted trial runs of the software.

<table>
<thead>
<tr>
<th>USER</th>
<th>Problem(s) if any</th>
<th>Menu Options</th>
<th>Interactive session</th>
<th>Overall assessment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>None</td>
<td>Well designed</td>
<td>Fun</td>
<td>Good</td>
</tr>
<tr>
<td>2</td>
<td>None</td>
<td>Captures attention</td>
<td>Straight forward</td>
<td>Very good</td>
</tr>
<tr>
<td>3</td>
<td>None</td>
<td>Very nice</td>
<td>Simple</td>
<td>B⁺</td>
</tr>
<tr>
<td>4</td>
<td>None</td>
<td>Easy to follow</td>
<td>Easy to follow</td>
<td>A</td>
</tr>
<tr>
<td>5</td>
<td>None</td>
<td>Colorful</td>
<td>Exciting</td>
<td>A</td>
</tr>
<tr>
<td>6</td>
<td>None</td>
<td>Simple and easy</td>
<td>Not complicated</td>
<td>Simple and fast results</td>
</tr>
<tr>
<td>7</td>
<td>None</td>
<td>It’s great</td>
<td>It’s simple</td>
<td>Relieves one of calculations</td>
</tr>
</tbody>
</table>
Problem #1. A vehicle, with all its wheels locked, leaves 40 feet (12.2 meters) skidmarks on the pavement, skids into a dirt and left 30 feet (9.15 meters) skidmarks before coming to a complete stop. Given that the coefficients of friction of the pavement and dirt are 0.7 and 0.5 respectively, what was the beginning speed at the start of the first skid? Assume grade as zero.

Book solution $= 35.9$ mph

$$V_i = \sqrt{V_e^2 + 30[s_1(f_1 + g) + s_2(f_2 + g)]}$$

where

- $V_i$ = initial speed, mph
- $V_e$ = final speed, mph
- $s_1$ = first skidmarks length, feet
- $s_2$ = second skidmarks length, feet
- $f_1$ = friction coefficient of pavement
- $f_2$ = friction coefficient of dirt
- $g$ = grade of pavement & dirt

$V_e = 0, \ s_1 = 40, \ f_1 = 0.7, \ g = 0, \ s_2 = 30, \ f_2 = 0.5$

Hence

$$V_i = \sqrt{30(40*0.7 + 30*0.5)}$$

---

\[ V_i = 35.9 \text{ mph} \]

Software requested input to solve the above problem:

- first skidmarks distance = 40 feet
- second skidmarks distance = 30 feet
- final speed = 0 mph
- grade of pavement and dirt = 0.0
- friction coefficient of pavement = 0.7
- friction coefficient of dirt = 0.5

Output of software solution:

- initial speed = 35.9 mph
- skidding time = 2.89 seconds
- deceleration = 18.3 ft/sec\(^2\)

Also Figures 7.15, 7.16 and 7.17 represents the graphical solution to the above problem. Tables 7.1, 7.2 and 7.3 are the corresponding tables to the graphs of Figures 7.15, 7.16 and 7.17 respectively.
Figure 7.15 Graph of Speed as a function of Braking Distance on two surfaces.
TABLE 7.2

friction coefficient of surface 1 = 0.70
friction coefficient of surface 2 = 0.50
grade of surfaces = 0.0

<table>
<thead>
<tr>
<th>Braking Distance (feet)</th>
<th>Surface 1 Speed (mph)</th>
<th>Surface 2 Speed (mph)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>35.9</td>
<td>--</td>
</tr>
<tr>
<td>5</td>
<td>34.4</td>
<td>--</td>
</tr>
<tr>
<td>10</td>
<td>32.9</td>
<td>--</td>
</tr>
<tr>
<td>15</td>
<td>31.2</td>
<td>--</td>
</tr>
<tr>
<td>20</td>
<td>29.5</td>
<td>--</td>
</tr>
<tr>
<td>25</td>
<td>27.7</td>
<td>--</td>
</tr>
<tr>
<td>30</td>
<td>25.7</td>
<td>--</td>
</tr>
<tr>
<td>35</td>
<td>23.6</td>
<td>--</td>
</tr>
<tr>
<td>40</td>
<td>21.2</td>
<td>--</td>
</tr>
<tr>
<td>45</td>
<td>--</td>
<td>19.3</td>
</tr>
<tr>
<td>50</td>
<td>--</td>
<td>17.3</td>
</tr>
<tr>
<td>55</td>
<td>--</td>
<td>15.0</td>
</tr>
<tr>
<td>60</td>
<td>--</td>
<td>12.2</td>
</tr>
<tr>
<td>65</td>
<td>--</td>
<td>8.6</td>
</tr>
<tr>
<td>70</td>
<td>--</td>
<td>0.0</td>
</tr>
</tbody>
</table>
Figure 7.16 Graph of Speed as a function of Braking Time on two different surfaces
TABLE 7.3

friction coefficient of surface 1 = 0.70
friction coefficient of surface 2 = 0.50
grade of surfaces = 0.0

<table>
<thead>
<tr>
<th>Braking Time (sec)</th>
<th>Braking Speed (mph)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td>35.9</td>
</tr>
<tr>
<td>0.50</td>
<td>28.2</td>
</tr>
<tr>
<td>1.00</td>
<td>20.7</td>
</tr>
<tr>
<td>1.50</td>
<td>15.2</td>
</tr>
<tr>
<td>2.00</td>
<td>9.7</td>
</tr>
<tr>
<td>2.50</td>
<td>4.2</td>
</tr>
<tr>
<td>2.88</td>
<td>0.0</td>
</tr>
</tbody>
</table>
Figure 7.17 Graph of Braking Distance as a function of Braking Time
**TABLE 7.4**

friction coefficient of surface 1 = 0.70  
friction coefficient of surface 2 = 0.50  
grade of surfaces = 0.0

<table>
<thead>
<tr>
<th>Braking Time (secs)</th>
<th>Braking Distance (feet)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td>0</td>
</tr>
<tr>
<td>0.05</td>
<td>18</td>
</tr>
<tr>
<td>1.00</td>
<td>40</td>
</tr>
<tr>
<td>1.50</td>
<td>42</td>
</tr>
<tr>
<td>2.00</td>
<td>49</td>
</tr>
<tr>
<td>2.50</td>
<td>59</td>
</tr>
<tr>
<td>2.89</td>
<td>70</td>
</tr>
</tbody>
</table>
Problem #2. An automobile leaves 22 feet (6.7 meters) of four-wheel skid on an asphalt street having a friction coefficient of 0.7, strikes a chuckhole and overturns, leaving an additional 31 feet (9.46 meters) of marks from the top in the asphalt (f=0.4), slides off the roadway and leaves 26 feet (7.93 meters) of overturned skid in the dirt (f=0.35). How fast was the automobile travelling at the start of the four-wheel skid? Grade is zero.

Book solution = 33.4 mph

\[ V_i = \sqrt{V_e^2 + 30[s_1(f_i g) + s_2(f_2 g) + s_3(f_3 g)]]} \]

\[ V_e = 0 \quad g = 0 \quad s_1 = 22 \quad f_i = 0.7 \]
\[ s_2 = 31 \quad f_2 = 0.4 \quad s_3 = 26 \quad f_3 = 0.35 \]

Hence

\[ V_i = \sqrt{30 [(22)(0.7) + (31)(0.4) + (26)(0.35)]} \]

\[ V_i = \sqrt{1107} \]

---

\[ V_i = 33.3 \text{ mph} \]

Software requested input to solve the above problem:

- first skidmarks distance = 22 feet
- second skidmarks distance = 31 feet
- third skidmarks distance = 26 feet
- final speed = 0 mph
- grade of roadway and dirt = 0.0
- friction coefficient of the roadway = 0.7
- friction coefficient of dirt = 0.5

Output of software solution:

- initial speed = 33.3 mph
- skidding time = 3.67 seconds
- deceleration = 13.3 ft/sec^2

Figures 7.18, 7.19 and 7.19 represents the graphical solution to the above problem. Tables 7.4, 7.5 and 7.6 are the corresponding tables to the graphs of figures 7.15, 7.16 and 7.17 respectively.
Figure 7.18 Graph of Speed as a function of Braking Time on three surfaces
TABLE 7.5

friction coefficient of surface 1 = 0.70
friction coefficient of surface 2 = 0.40
friction coefficient of surface 3 = 0.35
grade of surfaces = 0.0

<table>
<thead>
<tr>
<th>Braking Time (sec)</th>
<th>Braking Speed (mph)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td>33.3</td>
</tr>
<tr>
<td>0.50</td>
<td>25.6</td>
</tr>
<tr>
<td>1.00</td>
<td>21.1</td>
</tr>
<tr>
<td>1.50</td>
<td>16.7</td>
</tr>
<tr>
<td>2.00</td>
<td>12.8</td>
</tr>
<tr>
<td>2.50</td>
<td>9.0</td>
</tr>
<tr>
<td>3.00</td>
<td>5.1</td>
</tr>
<tr>
<td>3.50</td>
<td>1.3</td>
</tr>
<tr>
<td>3.67</td>
<td>0.0</td>
</tr>
</tbody>
</table>
Figure 7.19 Graph of Speed as a function of Braking Distance on three surfaces
TABLE 7.6

friction coefficient of surface 1 = 0.70
friction coefficient of surface 2 = 0.40
friction coefficient of surface 3 = 0.35
grade of surfaces = 0.0

<table>
<thead>
<tr>
<th>Braking Distance (feet)</th>
<th>Surface 1 Speed (mph)</th>
<th>Surface 2 Speed (mph)</th>
<th>Surface 3 Speed (mph)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>33.3</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>5</td>
<td>31.7</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>10</td>
<td>29.9</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>15</td>
<td>28.1</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>20</td>
<td>26.2</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>22</td>
<td>25.4</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>27</td>
<td>--</td>
<td>24.2</td>
<td>--</td>
</tr>
<tr>
<td>32</td>
<td>--</td>
<td>22.9</td>
<td>--</td>
</tr>
<tr>
<td>37</td>
<td>--</td>
<td>21.6</td>
<td>--</td>
</tr>
<tr>
<td>42</td>
<td>--</td>
<td>20.1</td>
<td>--</td>
</tr>
<tr>
<td>47</td>
<td>--</td>
<td>18.6</td>
<td>--</td>
</tr>
<tr>
<td>52</td>
<td>--</td>
<td>16.9</td>
<td>--</td>
</tr>
<tr>
<td>53</td>
<td>--</td>
<td>16.5</td>
<td>--</td>
</tr>
<tr>
<td>58</td>
<td>--</td>
<td>--</td>
<td>14.8</td>
</tr>
<tr>
<td>63</td>
<td>--</td>
<td>--</td>
<td>12.9</td>
</tr>
<tr>
<td>68</td>
<td>--</td>
<td>--</td>
<td>10.7</td>
</tr>
<tr>
<td>73</td>
<td>--</td>
<td>--</td>
<td>7.9</td>
</tr>
<tr>
<td>78</td>
<td>--</td>
<td>--</td>
<td>3.1</td>
</tr>
<tr>
<td>79</td>
<td>--</td>
<td>--</td>
<td>0.0</td>
</tr>
</tbody>
</table>
Figure 7.20 Graph of Braking Distance as function of Braking Time
TABLE 7.7

friction coefficient of surface 1 = 0.70  
friction coefficient of surface 2 = 0.40  
friction coefficient of surface 3 = 0.35  
grade of surfaces = 0.0

<table>
<thead>
<tr>
<th>Braking Time (secs)</th>
<th>Braking Distance (feet)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td>0</td>
</tr>
<tr>
<td>0.50</td>
<td>21</td>
</tr>
<tr>
<td>1.00</td>
<td>35</td>
</tr>
<tr>
<td>1.50</td>
<td>52</td>
</tr>
<tr>
<td>2.00</td>
<td>54</td>
</tr>
<tr>
<td>2.50</td>
<td>58</td>
</tr>
<tr>
<td>3.00</td>
<td>65</td>
</tr>
<tr>
<td>3.50</td>
<td>75</td>
</tr>
<tr>
<td>3.67</td>
<td>79</td>
</tr>
</tbody>
</table>
Problem #3. Suppose that in a test skid from an initial speed of 30 mph (48.3 km/hr), a vehicle slid to a stop in 40 feet (12.2 meters). What was the drag factor?. Grade is zero. How much time did it take to slide that distance?.

Book solution: 0.75, 1.8 secs.

\[ t = \frac{1.36s}{V_i - V_e} \]

where

- \( f \) = friction coefficient
- \( t \) = skidding time, seconds
- \( s \) = skidding distance, feet
- \( V_i \) = initial speed, mph
- \( V_e \) = final speed, mph
- \( g \) = grade

\[ f = \frac{30^2 - 0 - 30(40)(0)}{30(40)} \]

\[ f = 0.75 \]

---

and

\[ t = \frac{1.36(40)}{30 - 0} \]

\[ t = 1.8 \text{ seconds} \]

Software requested input to solve the above example:

- initial speed \( = 30 \text{ mph} \)
- final speed \( = 0 \text{ mph} \)
- grade of surface \( = 0.0 \)
- skidding distance \( = 40 \text{ feet} \)

Output of software solution:

- friction coefficient \( = 0.75 \)
- skidding time \( = 1.82 \text{ seconds} \)
- deceleration \( = 24.1 \text{ ft/sec}^2 \)

Figures 7.21, 7.22 and 7.23 are graphs plotted following the above output and tables 7.7, 7.8 and 7.9 are the tables of figures 7.21, 7.22 and 7.23 respectively. These graphs how the speed decreases with braking distance, how the speed decreases with time, and how the braking distance increases with braking time.
Figure 7.21 Graph of Braking Time as a function of Speed
\textbf{TABLE 7.8}

\textit{friction coefficient of surface 1} = 0.75  \\
\textit{grade of surfaces} = 0.0  \\
\textit{initial speed} = 30 \text{ mph}

<table>
<thead>
<tr>
<th>Braking Speed (mph)</th>
<th>Braking Time (secs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.82</td>
</tr>
<tr>
<td>5</td>
<td>1.51</td>
</tr>
<tr>
<td>10</td>
<td>1.21</td>
</tr>
<tr>
<td>15</td>
<td>0.91</td>
</tr>
<tr>
<td>20</td>
<td>0.61</td>
</tr>
<tr>
<td>25</td>
<td>0.30</td>
</tr>
<tr>
<td>30</td>
<td>0.00</td>
</tr>
</tbody>
</table>
**Figure 7.22** Graph of Braking Time as a function of Braking Distance
**TABLE 7.9**

*friction coefficient of surface 1 = 0.75  
grade of surfaces = 0.0*

<table>
<thead>
<tr>
<th>Braking Distance (feet)</th>
<th>Braking Time (secs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.00</td>
</tr>
<tr>
<td>5</td>
<td>0.64</td>
</tr>
<tr>
<td>10</td>
<td>0.91</td>
</tr>
<tr>
<td>15</td>
<td>1.11</td>
</tr>
<tr>
<td>20</td>
<td>1.29</td>
</tr>
<tr>
<td>25</td>
<td>1.44</td>
</tr>
<tr>
<td>30</td>
<td>1.58</td>
</tr>
<tr>
<td>35</td>
<td>1.70</td>
</tr>
<tr>
<td>40</td>
<td>1.82</td>
</tr>
</tbody>
</table>
Figure 7.23 Graph of Speed as a function of Braking Distance.
TABLE 7.10

*friction coefficient of surface* \( l = 0.75 \)

*grade of surfaces* = 0.0

<table>
<thead>
<tr>
<th>Braking Distance (feet)</th>
<th>Braking Speed (mph)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>30.00</td>
</tr>
<tr>
<td>5</td>
<td>28.06</td>
</tr>
<tr>
<td>10</td>
<td>25.98</td>
</tr>
<tr>
<td>15</td>
<td>23.72</td>
</tr>
<tr>
<td>20</td>
<td>21.21</td>
</tr>
<tr>
<td>25</td>
<td>18.37</td>
</tr>
<tr>
<td>30</td>
<td>15.00</td>
</tr>
<tr>
<td>35</td>
<td>10.61</td>
</tr>
<tr>
<td>40</td>
<td>0.00</td>
</tr>
</tbody>
</table>
Problem #4. Consider a vehicle traveling at 60 mph (96.6 km/hr) on a roadway with a coefficient of forward skidding friction of 0.40. If the grade is level, what is the braking distance to decelerate to 30 mph (48.3 km/hr)?

Book solution = 225 feet

\[ s = \frac{V_i^2 - V_e^2}{30(f+g)} \]

where

\[ s = \text{braking distance, feet} \]
\[ V_i = \text{initial speed of vehicle, mph} \]
\[ V_e = \text{final speed of vehicle, mph} \]
\[ g = \text{grade} \]

\[ s = \frac{(60^2 - 30^2)}{30(0.40 + 0.0)} \]

\[ s = 225 \text{ feet} \]

---

Software requested input to solve the above example:

- initial speed = 60 mph
- final speed = 30 mph
- grade of surface = 0.0
- friction coefficient = 0.40

Output of software solution:

- skidding distance = 225.0 feet
- skidding time = 3.41 seconds
- deceleration = 12.9 ft/sec²

Figures 7.24 and 7.25 are graphs plotted following the above output and tables 7.10 and 7.11 are the tables corresponding to the figures 7.24 and 7.25 respectively. These graphs show how the speed decreases with braking distance and how the speed decreases with time.
Figure 7.24 Graph of Braking Distance as a function of Speed of an automobile slowing to a speed of 30 mph at the end of braking.
**TABLE 7.11**

*friction coefficient of surface* $1 = 0.40$

*grade of surfaces* $= 0.0$

*initial speed* $= 60$ mph

*final speed* $= 30$ mph

<table>
<thead>
<tr>
<th>Braking Speed (mph)</th>
<th>Braking Distance (feet)</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>225.00</td>
</tr>
<tr>
<td>35</td>
<td>197.91</td>
</tr>
<tr>
<td>40</td>
<td>166.67</td>
</tr>
<tr>
<td>45</td>
<td>131.25</td>
</tr>
<tr>
<td>50</td>
<td>91.67</td>
</tr>
<tr>
<td>55</td>
<td>47.92</td>
</tr>
<tr>
<td>60</td>
<td>0.00</td>
</tr>
</tbody>
</table>
Figure 7.25 Graph of Braking Time as a function of Speed of an automobile slowing to a speed of 30 mph at the end of braking.
**TABLE 7.12**

friction coefficient of surface 1 = 0.40  
grade of surfaces = 0.0  
initial speed = 60 mph  
final speed = 30 mph

<table>
<thead>
<tr>
<th>Braking Speed (mph)</th>
<th>Braking Time (secs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>3.41</td>
</tr>
<tr>
<td>35</td>
<td>2.84</td>
</tr>
<tr>
<td>40</td>
<td>2.27</td>
</tr>
<tr>
<td>45</td>
<td>1.70</td>
</tr>
<tr>
<td>50</td>
<td>1.14</td>
</tr>
<tr>
<td>55</td>
<td>0.57</td>
</tr>
<tr>
<td>60</td>
<td>0.00</td>
</tr>
</tbody>
</table>
Figure 7.26 Graph of Braking Distance as a function of Braking Time
TABLE 7.13

\( \text{friction coefficient of surface} = 0.40 \)

\( \text{grade of surfaces} = 0.0 \)

\( \text{final speed} = 30 \text{ mph} \)

<table>
<thead>
<tr>
<th>Braking Time (sec)</th>
<th>Braking Distance (feet)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>0.50</td>
<td>23.61</td>
</tr>
<tr>
<td>1.00</td>
<td>50.45</td>
</tr>
<tr>
<td>1.50</td>
<td>80.50</td>
</tr>
<tr>
<td>2.00</td>
<td>113.78</td>
</tr>
<tr>
<td>2.50</td>
<td>150.27</td>
</tr>
<tr>
<td>3.00</td>
<td>189.99</td>
</tr>
<tr>
<td>3.41</td>
<td>225.00</td>
</tr>
</tbody>
</table>
Problem #5. A vehicle hits a bridge abutment at a speed estimated by investigation as 15 mph (24.2 km/hr). Skid marks of 100 feet (30.5 meters) on the pavement ($f_1=0.35$) followed by skid marks of 200 feet (61 meters) on the gravel shoulder approaching the abutment ($f_2=0.50$) are observed. The grade is level. What was the initial speed of the vehicle?

Book solution = 65.38 mph

$$V_i = \sqrt{V_e^2 + 30[s_1(f_1+g) + s_2(f_2+g)]}$$

where

$V_i$ = initial speed, mph
$V_e$ = final speed, mph
$s_1$ = skidmarks length on pavement, feet
$s_2$ = skidmarks length on gravel, feet
$f_1$ = coefficient of friction of pavement
$f_2$ = coefficient of friction of gravel
$g$ = grade

$$V_i = \sqrt{15^2 + 30[(100)(0.35) + (200)(0.50)]}$$

---

\[ V_i = 65.38 \text{ mph} \quad (105.3 \text{ km/hr}) \]

Software requested input to solve the above problem:

- first skidmarks distance = 100 feet
- second skidmarks distance = 200 feet
- final speed = 15 mph
- grade of pavement and gravel = 0.0
- friction coefficient of the pavement = 0.35
- friction coefficient of the gravel = 0.50

Output of software solution:

- initial speed = 65.4 mph
- skidding time = 4.92 seconds
- deceleration = 15.0 ft/sec²

Figures 7.26, 7.27 and 7.28 represents the graphical solution to the above problem. Tables 7.12, 7.13 and 7.14 are the corresponding tables to the graphs of figures 7.26, 7.27 and 7.28 respectively.

These graphs how the speed decreases with braking distance, how the speed decreases with time, and how the braking distance increases with braking time. From these figures and tables, it is easy to see the speed the vehicle was travelling or skidding just before it traverses into the gravel shoulder or the time it took to skid on the pavement and the time it took to skid on the gravel.
Figure 7.27 Graph of Speed as a function of Braking Distance of an automobile slowing to a speed of 15 mph at the end of braking on two different surfaces
TABLE 7.14

friction coefficient of surface 1 = 0.35
friction coefficient of surface 2 = 0.50
grade of surfaces = 0.0
final speed = 15 mph

<table>
<thead>
<tr>
<th>Braking Distance (feet)</th>
<th>Surface 1 Speed (mph)</th>
<th>Surface 2 Speed (mph)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>65.4</td>
<td>--</td>
</tr>
<tr>
<td>10</td>
<td>64.6</td>
<td>--</td>
</tr>
<tr>
<td>20</td>
<td>63.8</td>
<td>--</td>
</tr>
<tr>
<td>30</td>
<td>62.9</td>
<td>--</td>
</tr>
<tr>
<td>40</td>
<td>62.9</td>
<td>--</td>
</tr>
<tr>
<td>50</td>
<td>61.2</td>
<td>--</td>
</tr>
<tr>
<td>60</td>
<td>60.4</td>
<td>--</td>
</tr>
<tr>
<td>70</td>
<td>59.5</td>
<td>--</td>
</tr>
<tr>
<td>80</td>
<td>58.6</td>
<td>--</td>
</tr>
<tr>
<td>90</td>
<td>57.7</td>
<td>--</td>
</tr>
<tr>
<td>100</td>
<td>56.8</td>
<td>--</td>
</tr>
<tr>
<td>110</td>
<td>--</td>
<td>55.4</td>
</tr>
<tr>
<td>120</td>
<td>--</td>
<td>54.1</td>
</tr>
<tr>
<td>130</td>
<td>--</td>
<td>52.7</td>
</tr>
<tr>
<td>140</td>
<td>--</td>
<td>51.2</td>
</tr>
<tr>
<td>150</td>
<td>--</td>
<td>49.7</td>
</tr>
<tr>
<td>160</td>
<td>--</td>
<td>48.2</td>
</tr>
<tr>
<td>170</td>
<td>--</td>
<td>46.6</td>
</tr>
<tr>
<td>180</td>
<td>--</td>
<td>45.0</td>
</tr>
<tr>
<td>190</td>
<td>--</td>
<td>43.3</td>
</tr>
<tr>
<td>200</td>
<td>--</td>
<td>41.5</td>
</tr>
<tr>
<td>210</td>
<td>--</td>
<td>39.7</td>
</tr>
<tr>
<td>220</td>
<td>--</td>
<td>37.7</td>
</tr>
<tr>
<td>230</td>
<td>--</td>
<td>35.7</td>
</tr>
<tr>
<td>240</td>
<td>--</td>
<td>33.5</td>
</tr>
<tr>
<td>250</td>
<td>--</td>
<td>31.2</td>
</tr>
<tr>
<td>260</td>
<td>--</td>
<td>28.7</td>
</tr>
<tr>
<td>270</td>
<td>--</td>
<td>25.9</td>
</tr>
<tr>
<td>280</td>
<td>--</td>
<td>22.9</td>
</tr>
<tr>
<td>290</td>
<td>--</td>
<td>19.3</td>
</tr>
<tr>
<td>300</td>
<td>--</td>
<td>15.0</td>
</tr>
</tbody>
</table>
Figure 7.28 Graph of Speed as a function of Braking Time of an automobile slowing to a speed of 15 mph at the end of braking on two different surfaces
### TABLE 7.15

friction coefficient of surface 1 = 0.35  
friction coefficient of surface 2 = 0.50  
grade of surfaces = 0.0  
final speed = 15 mph

<table>
<thead>
<tr>
<th>Braking Time (sec)</th>
<th>Braking Speed (mph)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td>65.4</td>
</tr>
<tr>
<td>0.50</td>
<td>61.5</td>
</tr>
<tr>
<td>1.00</td>
<td>57.7</td>
</tr>
<tr>
<td>1.50</td>
<td>52.6</td>
</tr>
<tr>
<td>2.00</td>
<td>47.1</td>
</tr>
<tr>
<td>2.50</td>
<td>41.6</td>
</tr>
<tr>
<td>3.00</td>
<td>36.1</td>
</tr>
<tr>
<td>3.50</td>
<td>30.6</td>
</tr>
<tr>
<td>4.00</td>
<td>25.1</td>
</tr>
<tr>
<td>4.50</td>
<td>19.6</td>
</tr>
<tr>
<td>4.92</td>
<td>15.0</td>
</tr>
</tbody>
</table>
Figure 7.29 Graph of Braking Distance as a function of Braking Time of an automobile slowing to a speed of 15 mph at the end of braking on two different surfaces
TABLE 7.16

Friction coefficient of surface 1 = 0.35  
Friction coefficient of surface 2 = 0.50  
Grade of surfaces = 0.0  
Final speed = 15 mph

<table>
<thead>
<tr>
<th>Braking Time</th>
<th>Braking Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>(secs)</td>
<td>(feet)</td>
</tr>
<tr>
<td>0.00</td>
<td>0</td>
</tr>
<tr>
<td>0.50</td>
<td>43</td>
</tr>
<tr>
<td>1.00</td>
<td>89</td>
</tr>
<tr>
<td>1.50</td>
<td>110</td>
</tr>
<tr>
<td>2.00</td>
<td>126</td>
</tr>
<tr>
<td>2.50</td>
<td>146</td>
</tr>
<tr>
<td>3.00</td>
<td>170</td>
</tr>
<tr>
<td>3.50</td>
<td>198</td>
</tr>
<tr>
<td>4.00</td>
<td>230</td>
</tr>
<tr>
<td>4.50</td>
<td>267</td>
</tr>
<tr>
<td>4.92</td>
<td>300</td>
</tr>
</tbody>
</table>
Problem #6. A vehicle left two skidmarks on a clean, dry and well travelled pavement having a level grade in the direction of travel after turning almost end for end and came to a complete stop. (Indicates rear wheels sliding while front wheels rolling). The skidmarks distance was found to be 320 feet and the friction coefficient of the pavement given as 0.65. What is the initial speed of the vehicle. Since this was found to be irregular braking, the following information on the vehicle was obtained:

Vehicle wheelbase = 9.39 ft

weight of rear axle when car is level = 1425 lb.

weight of rear axle when front axle is hoisted h ft. = 1525 lb.

vehicle's total weight = 3175 lb

Height to which axle is lifted = 2.44 ft

Radius of vehicle tires = 0.99 ft

Height of center of mass as a decimal fraction of the wheelbase is

\[ x = \frac{(R/L) + (w_h - w_v) \sqrt{L^2 - h^2}}{hw} \]

---

\[ x = \frac{0.99/9.39}{2.44} + \frac{(1525-1425)}{2.44 \times 3175} \]

\[ x = 0.222 \]

Position of the center of mass behind the front axle is

\[ y_f = \frac{w_r}{w} \]

\[ y_f = \frac{1425}{3175} = 0.449 \]

The drag factor of the rear axle = 0.65

The drag factor of the front axle = 0.01 (rolling drag)

The total resultant drag factor of vehicle is

\[ f = \frac{f_r(1-y_f) + f_y y_f}{1-x(f_f-f_r)} \]

\[ f = \frac{0.01(1-0.449) + 0.65 \times 0.449}{1-0.222(0.01-0.65)} \]
\[ f = \frac{0.2974}{1.142} = 0.26 \]

\[ V_i = 30s(f+g) - V_e^2 \]

\[ V_i = \sqrt{30 \times 320(0.26+0)} - 0^2 \]

\[ V_i = 50.0 \text{ mph} \]

The software solution requires the following input values:

- vehicle wheelbase length = 9.39 feet
- weight of vehicle = 3175 lbs
- weight of rear axle when vehicle is level = 1425 lbs.
- weight of rear axle when the front axle is hoisted h feet = 1525 lbs.
- height to which the front axle is hoisted = 2.44 feet
- radius of vehicle tires = 0.99 feet
- skidding distance = 320 feet
- friction coefficient of pavement = 0.65
- friction of rolling wheels = 0.01
- final speed of the vehicle = 0 mph
- grade of the pavement = 0.0
The output from the software to problem #6:

- the height of the center of mass as a fraction of wheelbase = 0.22 feet
- position of the center of mass behind the front axle = 0.45
- the resultant drag factor required to stop the vehicle = 0.26
- the initial speed of the vehicle = 50.4 mph
- the skidding time of the vehicle = 8.67 seconds.
- the rate of deceleration of the vehicle = 8.5 ft/sec².

Figures 7.29, 7.30 and 7.31 represent the graphical solutions to the above problem. Tables 7.15, 7.16 and 7.17 are the corresponding tables to the graphs of figures 7.29, 7.30 and 7.31 respectively.

These graphs show how the speed decreases with braking distance, how the speed decreases with time, and how the braking distance increases with braking time. Had all the wheels locked during the braking, the resultant drag factor would have been 0.65 (same as the friction coefficient of the level pavement), and the vehicle would have been able to stop from 50 mph in feet for 3.5 seconds.
Figure 7.30 Graph of Speed as a function of Braking Distance. (A case in which only the rear wheels were locked and the front wheels rolling during the braking)
TABLE 7.17

friction coefficient of pavement = 0.65
rolling friction of front wheels = 0.01
grade of surfaces = 0.0

<table>
<thead>
<tr>
<th>Braking Distance (feet)</th>
<th>Speed (mph)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>50.4</td>
</tr>
<tr>
<td>10</td>
<td>49.6</td>
</tr>
<tr>
<td>20</td>
<td>47.9</td>
</tr>
<tr>
<td>30</td>
<td>47.1</td>
</tr>
<tr>
<td>40</td>
<td>46.3</td>
</tr>
<tr>
<td>50</td>
<td>45.4</td>
</tr>
<tr>
<td>60</td>
<td>44.5</td>
</tr>
<tr>
<td>70</td>
<td>43.6</td>
</tr>
<tr>
<td>80</td>
<td>42.7</td>
</tr>
<tr>
<td>90</td>
<td>41.8</td>
</tr>
<tr>
<td>100</td>
<td>40.8</td>
</tr>
<tr>
<td>110</td>
<td>39.8</td>
</tr>
<tr>
<td>120</td>
<td>38.8</td>
</tr>
<tr>
<td>130</td>
<td>37.8</td>
</tr>
<tr>
<td>140</td>
<td>36.7</td>
</tr>
<tr>
<td>150</td>
<td>35.6</td>
</tr>
<tr>
<td>160</td>
<td>34.5</td>
</tr>
<tr>
<td>170</td>
<td>33.3</td>
</tr>
<tr>
<td>180</td>
<td>32.1</td>
</tr>
<tr>
<td>190</td>
<td>30.8</td>
</tr>
<tr>
<td>200</td>
<td>29.5</td>
</tr>
<tr>
<td>210</td>
<td>28.2</td>
</tr>
<tr>
<td>220</td>
<td>26.7</td>
</tr>
<tr>
<td>230</td>
<td>25.2</td>
</tr>
<tr>
<td>240</td>
<td>23.6</td>
</tr>
<tr>
<td>250</td>
<td>21.8</td>
</tr>
<tr>
<td>260</td>
<td>19.9</td>
</tr>
<tr>
<td>270</td>
<td>17.8</td>
</tr>
<tr>
<td>280</td>
<td>15.4</td>
</tr>
<tr>
<td>290</td>
<td>12.6</td>
</tr>
<tr>
<td>300</td>
<td>8.6</td>
</tr>
<tr>
<td>310</td>
<td>0.8</td>
</tr>
<tr>
<td>320</td>
<td>0.0</td>
</tr>
</tbody>
</table>
Figure 7.31 Graph of Speed as a function of Braking Time of an automobile braking to a complete stop with only the rear wheels.
TABLE 7.18

friction coefficient of pavement = 0.65
rolling friction of front wheels = 0.01
grade of surfaces = 0.0

<table>
<thead>
<tr>
<th>Braking Time (secs)</th>
<th>Braking Speed (mph)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td>50.4</td>
</tr>
<tr>
<td>0.50</td>
<td>47.5</td>
</tr>
<tr>
<td>1.00</td>
<td>44.6</td>
</tr>
<tr>
<td>1.50</td>
<td>41.6</td>
</tr>
<tr>
<td>2.00</td>
<td>38.7</td>
</tr>
<tr>
<td>2.50</td>
<td>35.8</td>
</tr>
<tr>
<td>3.00</td>
<td>32.9</td>
</tr>
<tr>
<td>3.50</td>
<td>30.0</td>
</tr>
<tr>
<td>4.00</td>
<td>27.1</td>
</tr>
<tr>
<td>4.50</td>
<td>24.2</td>
</tr>
<tr>
<td>5.00</td>
<td>21.3</td>
</tr>
<tr>
<td>5.50</td>
<td>18.4</td>
</tr>
<tr>
<td>6.00</td>
<td>15.5</td>
</tr>
<tr>
<td>6.50</td>
<td>12.6</td>
</tr>
<tr>
<td>7.00</td>
<td>9.7</td>
</tr>
<tr>
<td>7.50</td>
<td>6.8</td>
</tr>
<tr>
<td>8.00</td>
<td>3.9</td>
</tr>
<tr>
<td>8.50</td>
<td>1.0</td>
</tr>
<tr>
<td>8.67</td>
<td>0.0</td>
</tr>
</tbody>
</table>
Figure 7.32 Graph of Braking Distance as a function of Braking Time of an automobile braking to a complete stop with only the rear wheels
TABLE 7.19

friction coefficient of pavement = 0.65
rolling friction of front wheels = 0.01
resultant drag factor = 0.26
grade of surfaces = 0.0

<table>
<thead>
<tr>
<th>Braking Distance (feet)</th>
<th>Braking Time (secs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.00</td>
</tr>
<tr>
<td>10</td>
<td>1.53</td>
</tr>
<tr>
<td>20</td>
<td>2.17</td>
</tr>
<tr>
<td>30</td>
<td>2.66</td>
</tr>
<tr>
<td>40</td>
<td>3.07</td>
</tr>
<tr>
<td>50</td>
<td>3.43</td>
</tr>
<tr>
<td>60</td>
<td>3.76</td>
</tr>
<tr>
<td>70</td>
<td>4.06</td>
</tr>
<tr>
<td>80</td>
<td>4.34</td>
</tr>
<tr>
<td>90</td>
<td>4.60</td>
</tr>
<tr>
<td>100</td>
<td>4.85</td>
</tr>
<tr>
<td>110</td>
<td>5.08</td>
</tr>
<tr>
<td>120</td>
<td>5.31</td>
</tr>
<tr>
<td>130</td>
<td>5.53</td>
</tr>
<tr>
<td>140</td>
<td>5.74</td>
</tr>
<tr>
<td>150</td>
<td>5.94</td>
</tr>
<tr>
<td>160</td>
<td>6.13</td>
</tr>
<tr>
<td>170</td>
<td>6.32</td>
</tr>
<tr>
<td>180</td>
<td>6.50</td>
</tr>
<tr>
<td>190</td>
<td>6.68</td>
</tr>
<tr>
<td>200</td>
<td>6.86</td>
</tr>
<tr>
<td>210</td>
<td>7.03</td>
</tr>
<tr>
<td>220</td>
<td>7.19</td>
</tr>
<tr>
<td>230</td>
<td>7.35</td>
</tr>
<tr>
<td>240</td>
<td>7.51</td>
</tr>
<tr>
<td>250</td>
<td>7.67</td>
</tr>
<tr>
<td>260</td>
<td>7.82</td>
</tr>
<tr>
<td>270</td>
<td>7.97</td>
</tr>
<tr>
<td>280</td>
<td>8.11</td>
</tr>
<tr>
<td>290</td>
<td>8.26</td>
</tr>
<tr>
<td>300</td>
<td>8.40</td>
</tr>
<tr>
<td>310</td>
<td>8.54</td>
</tr>
<tr>
<td>320</td>
<td>8.67</td>
</tr>
</tbody>
</table>
QUESTIONNAIRE DESIGNED FOR USER FEEDBACK AFTER USING
SKIDPRO

Date: ______________________
Name: ______________________
Address: ______________________
City/Town ______________________ State ________ Zip ________
Educational background ______________________
Company (if applicable) ______________________

1. What problem(s)(if any) did you encounter working with the Software (SKIDPRO)?.
   ______________________

2. Were the menu options self-explanatory relative to what you were looking to find?.
   ______________________

3. Did the screen display help in terms of readability and information digestion?.
   ______________________

4. Are the questions you encountered in the interactive session structured to your understanding?.
   ______________________
5. What is the thing(s) you like or dislike about SKIDPRO?

6. What is your overall assessment of the software?

7. Is the interactive process clear/vague/in-between?

8. What problem(s) did you see in using the traditional method?

9. If formulas were not provided, would you have been able to solve the problems using the traditional method?

10. Which approach would you be comfortable with, the traditional way or using the Software Package, and why?
7.8 Program Documentation and a User’s Guide

The software package is written in Turbo C language. The software is designed to run in either the USA units or the Metric units. The program under each units option is split into two because they are too large to compile in the Turbo C environment. The parts are compiled separately and linked during the compilation of the main program. The program source code is listed in Appendix A. Each of the two units option program has approximately 5000 lines of code. Finally, a users guide for use of the software package is included in Appendix F.
CHAPTER 8

CONCLUSIONS AND RECOMMENDATIONS

Automobile skidding is a function of the driver, speed, road surface, brake efficiency and a host of other factors some of which are not very significant. Translating these factors into appropriate standards and criteria are important design considerations. Since the driver is recognized as a dominant factor in skidding accidents, analysis of the driver-vehicle highway relationship in skidding accidents should be geared towards educating the driver on the relationships between the most pertinent factors involved in vehicle skidding, namely speed, braking distance, coefficient of friction, road elevation etc.. Research has been focused on the human factors aspect of vehicle and the roadway design without much emphasis on the 'driver factor'. Steps should be taken to educate drivers on vehicle and roadway design standards and how they can translate or relate to such standards when confronted with a decision to maneuver a moving vehicle to avoid a possible accident.

The software is developed to provide precise values of such parameters as speed, braking distance, coefficient of friction, and braking time, all of which are inputs to an accident investigation. It generates these values based on information from the user or accident investigator about a vehicle state prior to braking and/or after braking. The output results from the software are found to be very accurate. It thus fulfills its objective of generating precise values of the parameters mentioned above. In addition to this, the software generates various graphs and tables from which accurate
values of these parameters can be obtained. It’s menu driven environment allows the user to work with sections or modules of the program one at a time but can also call on other modules for some information. The user can repeat a section or module as many times as desired before exiting from the entire program or moving to another module. The 'Calculator' module of the software is designed to work like a calculator. It gives the user quick and precise output without going through much interrogative question session. Perhaps the most remarkable feature of the Package is its units option. This option makes the software more unique by its ability to operate in either the English units environment or the Metric units environment. This gives it more flexibility and makes it more usable in the International arena.

The information generated from the software will not only assist accident investigators, but the Police department, Transportation department, Insurance companies, Institutions of higher learning and the public at large at a better understanding of the causes, prevention and control of skids. This can be achieved through launching a widespread driver education involving all public agencies concerned with highway safety. Even though the software covers a broader spectrum of vehicle-road scenarios, it does not cover vehicle collision cases in terms of impact nature, force or damage extent. It also addresses only horizontal curvatures and does not address vehicle skidding on curved pavements. These other areas can be taken up in other thesis project in the near future.
The major conclusions in this thesis are:

1. The software package was successfully developed, works, easy to use, accurate and validated.

2. The software package should be easy to use by Police, etc. in the field using laptop.

3. It can be of potential use in driver education.

4. It can be used with touch screen and windows to let people select from pictures.

User input values for speed, distance, time and friction coefficient has been restricted to certain minimum and maximum values to fit the limits of the graphs generated. The software can be further improved or updated to make adjustable scale drafts for any input values of the parameters mentioned above. Finally, more research should be made on how much drivers really know about the factors involved in skidding.
BIBLIOGRAPHY


43. D.L. Woods, *Sensitivity Analysis of the Factors Affecting Highway Vertical*


<table>
<thead>
<tr>
<th>CONTENTS OF PROGRAM LISTING</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>USA.PRJ</td>
<td>A-1</td>
</tr>
<tr>
<td>USA.H</td>
<td>A-2</td>
</tr>
<tr>
<td>UNIT.BAT</td>
<td>A-4</td>
</tr>
<tr>
<td>USA1.C</td>
<td>A-6</td>
</tr>
<tr>
<td>USA2.C</td>
<td>A-35</td>
</tr>
<tr>
<td>METRIC1.C</td>
<td>A-66</td>
</tr>
<tr>
<td>METRIC2.C</td>
<td>A-94</td>
</tr>
<tr>
<td>METRIC1.H</td>
<td>A-125</td>
</tr>
<tr>
<td>TITLE.C</td>
<td>A-127</td>
</tr>
</tbody>
</table>
USA.PRJ

usa1.c
usa2.c
graphics.lib
USA.H

#define Vs_min1(Ve_min, s1_min, dmin_1, g, wv, wt) sqrt(30*s1_min* ((wv/ (wv+wt)) *dmin_1 +g) + Ve_min*Ve_min)

#define Vs_max1(Ve_max, s1_max, dmax_1, g, wv, wt) sqrt(30*s1_max* ((wv/(wv+wt)) *dmax_1 +g) + Ve_max*Ve_max)

#define Vs_max2(Ve_min, s1_min, dmin_1, g, wv, wt) sqrt(30*s1_min* (dmin_1 + g) + Ve_min*Ve_min)

#define Vs_max2(Ve_max, s1_max, dmax_1, g, wv, wt) sqrt(30*s1_max* (dmax_1 + g) + Ve_max*Ve_max)

#define Vs_max3(Ve_max, s1_max, s2_max, dmax_1, dmax_2, g, wv, wt) sqrt(30* ((wv/(wv + wt)) *(s1_max*dmax_1 +s2_max*dmax_2) +g*(s1_max+s2_max)) +Ve_max*Ve_max)

#define Vs_max4(Ve_max, s1_max, s2_max, dmax_1, dmax_2, g) sqrt(30* ((s1_max * dmax_1 + s2_max * dmax_2) +g*(s1_max+s2_max)) +Ve_max*Ve_max)

#define Vs_max5(Ve_max, s1_max, s2_max, dmax_1, dmax_2, g) sqrt(15 *s1_max * (dmax_1 + dmax_2) +g*(s1_max+s2_max) +Ve_max*Ve_max)

#define Vs_max6(Ve_max, s1_max, s2_max, dmax_1, dmax_2, g) sqrt(15 *s1_max * ((dmax_1 + dmax_2) + 2*g) +Ve_max*Ve_max)

#define Vs_max7(Ve_max, s1_max, s2_max, s3_max, dmax_1, dmax_2, dmax_3, g, wv, wt) sqrt(30* ((wv/(wv + wt)) *(s1_max*dmax_1 + s2_max*dmax_2 + s3_max*dmax_3) + g*(s1_max+s2_max+s3_max)) +Ve_max*Ve_max)

#define Vs_max8(Ve_max, s1_max, s2_max, s3_max, dmax_1, dmax_2, dmax_3, g) sqrt(30* ((s1_max * dmax_1 + s2_max * dmax_2 + s3_max*dmax_3) +g*(s1_max+s2_max+s3_max)) +Ve_max*Ve_max)

#define Vs_max9(Ve_max, s1_max, dmax_1, dmax_2, dmax_3, g, wv, wt) sqrt(10 *s1_max *((dmax_1 + dmax_2 +dmax_3) +wv/(wv+wt)) +3*g) +Ve_max*Ve_max)

#define Vs_max10(Ve_min, s1_max, dmax_1, dmax_2, dmax_3, g) sqrt(10 *s1_max * (dmax_1 + dmax_2+dmax_3+g) +Ve_max*Ve_max)

#define Vs_max11(Ve_max, s1_max, dmax_1, dmax_2, dmax_3, g, wv, wt) sqrt(30 *s1_max * ((wv/(wv+wt)) *dmax +g) + Ve_max*Ve_max)

#define Vs_max12(Ve_max, s1_max, dmax_1, dmax_2, dmax_3, g) sqrt(30 *s1_max * (dmax +g) + Ve_max*Ve_max)

#define Vi_max(Vs_max, Ve_max) sqrt((Vs_max*Vs_max)+(Ve_max*Ve_max))

#define Ve_max(Vs_max, Vi_max) sqrt((Vi_max*Vi_max)-(Vs_max*Vs_max))

#define t_max(Vs_max, Ve_max, dmax) (Vs_max-Ve_max)/(22*(dmax +g))
#define sl_maxl(Vs_max, Ve_max, dmax, g, wv, wt) ((Vs_max * Vs_max) - (Ve_max * Ve_max)) / (30*(wv/(wv+wt)) + dmax + g))

#define sl_max2(Vs_max, Ve_max, dmax, g) ((Vs_max * Vs_max) - (Ve_max * Ve_max)) / (30*(dmax + g))

#define fmax_1(Vs_max, Ve_max, sl_max, g, wv, wt) ((Vs_max * Vs_max) - (Ve_max * Ve_max) + (30*sl_max*g))/(30*sl_max*(wv/(wv+wt)))

#define fmax_2(Vs_max, Ve_max, sl_max, g) ((Vs_max * Vs_max) - (Ve_max * Ve_max) + (30*sl_max*g))/(30*sl_max)

#define pd_max(Vs_max, pt_max) 1.468*Vs_max*pt_max

#define ts_max(Vs_max, pt_max, dmax, g) (1.468* Vs_max * pt_max) + ((Vs_max * Vs_max) / (30*(dmax + g)))

#define sec(Vs_max, dmax_1, g) (0.0455*Vs_max/(dmax_1 + g))

#define sec1(Vs_max, dmax_2, g) (0.0455*Vs_max/(dmax_2 + g))

#define sec2(Vs_max, dmax_3, g) (0.0455*Vs_max/(dmax_3 + g))

#define f1(vs, s) 0.0334*(vs*vs)/s

#define f2(vs, t) 0.0455*vs/t

#define f3(s, t) 0.0621*s/(t*t)

#define s1(vs, f) 0.0334*vs*vs/f

#define s2(vs, t) 0.733*vs*t

#define s3(f, t) 16.1*f*t*t

#define t2(vs, f) 0.0455*vs/f

#define t3(s, f) 0.249*sqrt(s/f)

#define t1(vs, s) 1.36*s/vs

#define vs1(s, f) 5.47*sqrt(s*f)

#define vs2(s, t) 1.36*s/t

#define vs3(f, t) 22*f*t
UNITS.BAT

@ECHO OFF
:MAIN
cls
nocurs
T1.EXE
ECHO
ECHO
ECHO
ECHO
ECHO
ECHO
ECHO
ECHO
ECHO
ECHO
ECHO
ECHO
ECHO
ECHO
ECHO
reply
if errorlevel 62 goto REPLY
if errorlevel 61 goto F3
if errorlevel 60 goto F2
if errorlevel 59 goto F1
goto REPLY
:F2
cls
normcurs
IF NOT EXIST USA.EXE GOTO F2N
ECHO Loading USA/English Please wait...
USA.EXE
goto MAIN
:F2N
ECHO You are missing USA.EXE
ECHO Please rerun the INSTALL program to rectify this problem
PAUSE
GOTO MAIN
:F3
cls
normcurs

IF NOT EXIST METRIC.EXE GOTO F3N
ECHO Loading Metric Please wait...
METRIC.EXE
goto MAIN
:F3N
ECHO You are missing METRIC.EXE
ECHO Please rerun the INSTALL program to rectify this problem
PAUSE
GOTO MAIN
normcurs
echo = = = = = EXITING SKIDPRO = = = = =
:F1
normcurs
A User Friendly Program to Determine Speed, Skidding Time, Skidding Distance and Friction Coefficient during an Automobile braking.

Implemented in Turbo C for MS-DOS V5.0 or newer version

prepared by A. S. Nuruddin as thesis for the degree of the Master of Science

Copyright 1993 H.T. Zwahlen, Ph.D.; A.S. Nuruddin; Ohio University
Department of Industrial and Systems Engineering
Ohio University, Athens, Ohio 45701-2979
November 1993

The following programs are needed to compile this software package.

USA1.C USA2.C USA.PRJ
USA.H UNIT.BAT

These programs must be pre-compiled and linked
do
{

textcolor(BLUE);
textbackground(LIGHTGRAY);
crsc();
gotoxy(23,4); cprintf("What parameter do you want to find? ");
gotoxy(10,5); cprintf("Select one of the following by typing the corresponding letter.");
textcolor(14);
textbackground(BLUE);
gotoxy(18,8); cprintf(" **** MAIN MENU ****");
gotoxy(18,9); cprintf(" A Initial or Final speed ");
gotoxy(18,10); cprintf(" B Skidding distance ");
gotoxy(18,11); cprintf(" C Skidding time ");
gotoxy(18,12); cprintf(" D Friction coefficient ");
gotoxy(18,13); cprintf(" E Calculator ");
gotoxy(18,14); cprintf(" Q Quit ");
gotoxy(18,15); cprintf(" ");
gotoxy(18,16); cprintf(" ");
textbackground(LIGHTGRAY);
/* Get and act on user input */
textcolor(14);
textbackground(MAGENTA);
gotoxy(18,18); cprintf(" Your selection is (A, B, C, D, E or Q) = ");
gotoxy(62,18); cprintf(" ");
textcolor(BLACK);
selection = toupper(getche());
crsc();
switch (selection)
{

case 'A' :
    speed();
    break;

case 'B' :
    skid_distance(&Vs_max, &fmax_l, &g, &s1_max, &Ve_max, &time);
    break;

case 'C' :
    skid_time(&Vs_max, &fmax_l, &g, &s1_max, &Ve_max, &time);
    break;

case 'D' :
    coeff_friction(&Vs_max, &fmax_l, &g, &s1_max, &Ve_max, &time);
    break;

case 'E' :
    calculator();
    break;

case 'Q' :
    exit(0);
        /* end of switch. */
    crsc();
}
    /* end of do */
while ((selection != 'A') &&(selection != 'B') &&(selection != 'C') &&(selection != 'D') &&(selection != 'E') &&(selection != 'Q'));
    /* End of Mainmenu */
void speed()
{
    char selection;
    clrscr();
    do
    {
        textcolor(BLUE);
        gotoxy(21,4); cprintf("Which speed type do you want to find? ");
        gotoxy(10,5); cprintf("Select one of the following by typing the corresponding letter");
        textbackground(BLUE);
        textcolor(14);
        gotoxy(18,8); cprintf("** SPEED MENU **");
        gotoxy(18,9); cprintf(" A] Initial_Speed");
        gotoxy(18,10); cprintf(" B] Final_Speed");
        gotoxy(18,11); cprintf(" C] Main menu");
        gotoxy(18,12); cprintf(" Q] Quit");
        gotoxy(18,13); cprintf(" Your selection is (A, B, C or Q) = > ");
        textbackground(MAGENTA);
        textcolor(YELLOW);
        gotoxy(18,16); cprintf(" Your selection is (A, B, C or Q) = >");
        gotoxy(58,16);
        textcolor(14);
        selection = toupper(getche());
        clrscr();
        switch (selection)
        {
            case 'A' : initial_speed(&Vs_max, &dmax_1, &dmax_2, &dmax_3, &s1_max, &s2_max, &s3_max, &g, &Ve_max, &timel, &time2, &time3,&curve);
                break;
            case 'B' : final_speed();
                break;
            case 'C' : mainmenu();
                break;
            case 'Q' : exit(0);
                press_return();
                clrscr();
        } /* end of switch */
    } /* end of do loop */
    while ((selection != 'A') && (selection != 'B') && (selection != 'C') && (selection != 'Q'));
}
void road_type()
{
  char selection;
  clrscr();
  do
  {
    textcolor(BLUE);
    gotoxy(16,4); cprintf("On how many different surfaces did skidding occur? ");
    gotoxy(10,5); cprintf("Select one of the following by typing the corresponding letter");
    textcolor(BLUE);
    gotoxy(18,8); cprintf("** ROAD SURFACE **");
    gotoxy(18,9); cprintf("A] One surface");
    gotoxy(18,11); cprintf("B] Two different surfaces");
    gotoxy(18,12); cprintf("C] Three different surfaces");
    gotoxy(18,13); cprintf("D] Return to Speed menu");
  }
  textbackground(MAGENTA);
  textcolor(14);
  gotoxy(18,16); cprintf(" Your selection is (A, B, C or D) = > ");
  gotoxy(10,17); cprintf(" ");
  textcolor(BLACK);
  gotoxy(58, 16);
  selection = toupper(getche());
  textbackground(LIGHTGRAY);
  getch();
  clrscr();

  switch (selection)
  {
    case 'A' :
      one_lane(&Vs_max, &dmax_1, &g, &s1_max, &Ve_max, &time1, &curve);
      break;
    case 'B' :
      two_lane(&Vs_max, &dmax_1, &dmax_2, &s1_max, &s2_max, &g, &Ve_max, &time1, &time2, &curve);
      break;
    case 'C' :
      three_lane(&Vs_max, &dmax_1, &dmax_2, &dmax_3, &s1_max, &s2_max, &s3_max, &g, &Ve_max, &time1, &time2, &time3, &curve);
      break;
    case 'D' :
      speed();
      break;
  } /* end of switch */
} /* end of do */

while ((selection != 'A') && (selection != 'B') && (selection != 'C') && (selection != 'D'));
char road_structure()
{
    char selection;
    clrscr();
    do
    {
        textcolor(BLUE);
        gotoxy(23,4); cprintf("Describe the grade of the road.");
        gotoxy(10,5); cprintf("Select one of the following by typing the corresponding letter");
        textbackground(BLUE);
        textcolor(14);
        gotoxy(18,8); cprintf("**** GRADE OF ROAD ****");
        gotoxy(18,10); cprintf(" ");
        gotoxy(18,11); cprintf(" A] Level ");
        gotoxy(18,12); cprintf(" B] Downhill ");
        gotoxy(18,13); cprintf(" C] Uphill ");
        gotoxy(18,14); cprintf(" D] Return to Road Surface ");
        gotoxy(18,15); cprintf(" ");
        textbackground(MAGENTA);
        textcolor(14);
        gotoxy(18,16); cprintf(" Your selection is (A, B, C or D) = > ");
        gotoxy(18,17); cprintf(" ");
        textcolor(BLACK);
        gotoxy(58,16);
        selection = toupper(getche());
        textbackground(LIGHTGRAY);
        getch();
        clrscr();
        switch(selection)
        {
            case 'A' :  g = 0.0;
                        break;
            case 'B' :  downhill();
                        break;
            case 'C' :  uphill();
                        break;
            case 'D' :  road_type();

        } /* end of switch */
    } /* end of do */
    while ((selection != 'A') && (selection != 'B') && (selection != 'C') && (selection != 'D'));
}
textcolor(BLUE);
gotoxy(23,15); cprintf("Your choice is (1, 2 or 3) = > ");

gotoxy(23,15); cprintf("Your choice is (1, 2 or 3) = > ");

textcolor(BLUE);
gets(user_choice);
choice = atoi(user_choice);
clrscr();
if (choice == 1) {
    strcpy(YLABEL, "SPEED - Miles per hour");
    strcpy(XLABEL, "BRAKING DISTANCE - Feet");
    xmin=0.0;
    yy=Vs_max;
    for (itemp=1; itemp <= curve; itemp++) {
        if (itemp == 1) {
            num=1;
            xmaximum[itemp] =s1_max;
        } else if (itemp == 2) {
            num=16;
            xmaximum[itemp] =s2_max +s1_max;
        } else if (itemp == 3) {
            num=17;
            xmaximum[itemp] = s3_max + s1_max + s2_max;
        }
    }
    setgraph(xmin, xmaximum, xma, yma, &maxx, &maxy, yy, num, curve);
} else
    if (choice == 2) {
        strcpy(YLABEL, "SPEED - Miles per hour");
        strcpy(XLABEL, "BRAKING TIME - Seconds");
        xmin=0.0;
        for (itemp=1; itemp <= curve; itemp++) {
            if (itemp == 1) {
                num=2;
                xmaximum[itemp] =time1;
            } else
                if (itemp == 2) {
                    num=24;
                    xmaximum[itemp] =time2 +time1;
                } else
                    if (itemp == 3) {
                        num=25;
                        xmaximum[itemp] =time3 +time2 +time1;
                    }
        }
        setgraph(xmin, xmaximum, xma, yma, &maxx, &maxy, yy, num, curve);
    }
else
if (choice == 3) {
    strcpy(XLABEL, "BRAKING TIME - Seconds");
    strcpy(YLABEL, "BRAKING DISTANCE - Feet");
    xmin = 0.0;
    for (itemp = 1; itemp <= curve; itemp++) {
        if (itemp == 1) {
            num = 3;
            xmaximum[itemp] = time1;
        } else
        if (itemp == 2) {
            num = 26;
            xmaximum[itemp] = time1 + time2;
        } else
        if (itemp == 3) {
            num = 27;
            xmaximum[itemp] = time1 + time2 + time3;
        }
    }
    setgraph(xmin, xmaximum, xma, yma, &maxx, &maxy, yy, num, curve);
} while ((choice != 1) && (choice != 2) && (choice != 3));
clrscr();
else
if (answer == 'N') {
    repeat = FALSE;
    clrscr();
gotoxy(24, 13); printf("This ends the session on initial speed.");
textcolor(RED);
gotoxy(18, 14); cprintf("Hit the ENTER/RETURN key to return to Main Menu");
getchar();
mainmenu();
} while ((answer != 'Y') && (answer != 'N'));
clrscr();
gotoxy(24, 13); printf("This ends the session on initial speed.");
textcolor(RED);
gotoxy(18, 14); cprintf("Hit the ENTER/RETURN key to return to Main Menu.");
getchar();
mainmenu();
A void final_speed()
{
    char selection, user_choice[2];
    float result, yy, dece;
    int repeat, curve, j;
    int choice, maxx, maxy, itemp;
    char answer, ch;
    char user_curve[2];
    float xmaximum[4];

clrscr();
road_structure();
textcolor(YELLOW);
gotoxy(20,12); cprintf("Enter the initial speed of the vehicle (mph) = > ");
Vs_max = Get_input(5.0, 200.0);
clrscr();
do 
{
    textcolor(YELLOW);
gotoxy(14,12); cprintf("How many different surfaces are involved. Chose number below");
textcolor(BLUE);
gotoxy(30,14); cprintf("1] one surface");
gotoxy(30,15); cprintf("2] two surfaces");
gotoxy(30,16); cprintf("3] three surfaces");
textcolor(YELLOW);
gotoxy(26,18); cprintf("Your choice is (1, 2, or 3) = > ");
textcolor(BLUE);
gets(user_choice);
choice=atoi(user_choice);
clrscr();
if (choice == 1) 
{
    curve=1;
    gotoxy(20,12); printf("Enter the coefficient of friction of the surface = > ");
dmax_1 =Get_input(0.01, 2.00);
gotoxy(20,14); printf("Enter the skidmarks length on the surface (feet) = > ");
s1_max =Get_input(1.0, 2000.0);
clrscr();
Ve1=sqrt((Vs_max*Vs_max)-30*s1_max*(dmax_1 + g));
time1 = 0.0455*(Vs_max-Ve1)/(dmax_1 + g);
  dece=(dmax_1 + g)*32.2;
gotoxy(15,13); cprintf("The final speed of the vehicle = > %2.1f mph", Ve1);
gotoxy(15,14); cprintf("The total skidding time = > %2.2f sec.", time1);
gotoxy(15,15); cprintf("The rate of slowing or deceleration = > %2.1f ft/sec/sec.", dece);
  press_return();
}
else if(choicen= =2) 
{
    curve=2;
    gotoxy(20,12); printf("Enter the coefficient of friction of surface_1 = > ");
dmax_1 =Get_input(0.01, 2.00);
gotoxy(20,14); printf("Enter the coefficient of friction of surface_2 = > ");
}
dmax_2 = Get_input(0.01, 2.00);
c1sr();
gotoxy(20,12); printf("Enter the skidmarks length on surface_1 (feet) = > ");
s1_max = Get_input(1.0, 2000.0);
gotoxy(20,14); printf("Enter the skidmarks length on surface_2 (feet) = > ");
s2_max = Get_input(1.0, 2000.0);
c1sr();
Vel = sqrt(Vs_max*Vs_max-30*sl_max*(dmax_1 + g));
Ve_max = sqrt(Ve1*Ve1-30*s2_max*(dmax_2 + g));
time1 = 0.0455*(Vs_max-Ve1)/(dmax_1 + g);
time2 = 0.0455*(Ve1-Ve_max)/(dmax_2 + g);
time = time1 + time2;
dece = 1.467*(Vs_max-Ve_max)/time;
gotoxy(15,13); cprintf("The final speed of the vehicle => %2.1f mph", Ve_max);
gotoxy(15,14); cprintf("The total skidding time => %2.2f sec.",time);
gotoxy(15,15); cprintf("The rate of slowing or deceleration => %2.1f ft/sec/sec.",dece);
getchar();
press _return();
}
else if(choice == 3) {
    textcolor(BLUE);
curve=3;
gotoxy(20,12); cprintf("Enter the coefficient of friction of surface_1 = > ");
dmax_1 = Get_input(0.01, 2.00);
gotoxy(20,14); cprintf("Enter the coefficient of friction of surface_2 = > ");
dmax_2 = Get_input(0.01, 2.00);
gotoxy(20,16); cprintf("Enter the coefficient of friction of surface_3 = > ");
dmax_3 = Get_input(0.01, 2.00);
c1sr();
gotoxy(20,12); cprintf("Enter the skidmarks length on surface_1 (feet) = > ");
s1_max = Get_input(1.0, 2000.0);
gotoxy(20,14); cprintf("Enter the skidmarks length on surface_2 (feet) = > ");
s2_max = Get_input(1.0, 2000.0);
gotoxy(20,16); cprintf("Enter the skidmarks length on surface_3 (feet) = > ");
s3_max = Get_input(1.0, 2000.0);
c1sr();
Ve1 = sqrt(Vs_max*Vs_max-30*s1_max*(dmax_1 + g));
Ve_max = sqrt(Ve1*Ve1-30*s2_max*(dmax_2 + g));
Ve2 = sqrt(Ve_max*Ve_max-30*s3_max*(dmax_3 + g));
time1 = 0.0455*(Vs_max-Ve1)/(dmax_1 + g);
time2 = 0.0455*(Ve1-Ve_max)/(dmax_2 + g);
time3 = 0.0455*(Ve_max-Ve2)/(dmax_3 + g);
time = time1 + time2 + time3;
dece = 1.467*(Vs_max-Ve2)/time;
gotoxy(15,13); cprintf("The final speed the vehicle = > %2.1f mph", Ve_max);
gotoxy(15,14); cprintf("The total skidding time = > %2.2f sec.",time);
gotoxy(15,15); cprintf("The rate of slowing or deceleration = > %2.1f ft/sec/sec.",dece);
getchar();
press _return();
while ((choice != 1) && (choice != 2) && (choice != 3));

/* Drawing graphs of final speed versus other parameters */  
repeat = TRUE;
while (repeat)
{
    do
    {
        textcolor(YELLOW);
        gotoxy(12,8); cprintf("Do you want to plot the speed profile? (Y/N) = > ");
        textcolor(BLUE);
        answer = toupper(getche());
        getch();
        clrscr();
        if (answer == 'Y')
        {
            textcolor(YELLOW);
            do
            {
                gotoxy(20,10); cprintf("Chose the type of graph you want to plot");
                gotoxy(20,12); printf("1] Speed as a function of distance");
                gotoxy(20,13); printf("2] Speed as a function of time");
                gotoxy(20,14); printf("3] Time as a function of distance");
                gotoxy(23,16); cprintf("Your choice is (1, 2 or 3) = > ");
                textcolor(BLUE);
                gets(user_choice);
                choice = atoi(user_choice);
                clrscr();
                if (choice == 1) {
                    strcpy(YLABEL, "SPEED - Miles per hour");
                    strcpy(XLABEL, "BRAKING DISTANCE - feet");
                    xmin = 0.0;
                    yy = Vs_max;
                    num = 18;
                    for (itemp = 1; itemp <= curve; itemp++) {
                        if (itemp == 1) {
                            num = 1;
                            xmaximum[itemp] = s1_max;
                        } else if (itemp == 2) {
                            num = 16;
                            xmaximum[itemp] = s2_max + s1_max;
                        } else if (itemp == 3) {
                            num = 17;
                            xmaximum[itemp] = s1_max + s2_max + s3_max;
                        }
                    }
                    setgraph(xmin, xmaximum, xma, yma, &maxx, &maxy, yy, num, curve);
                }
            } else
if (choice == 2) {
    strcpy(YLABEL, "SPEED - Miles per hour");
    strcpy(XLABEL, "TIME REQUIRED TO STOP - Seconds");
    xmin=0;
    for (j = 1; j <= curve; j++) {
        if (j == 1) {
            num=2;
            xmaximum[j]=time1;
        }
        else
            if (j == 2) {
                num=24;
                xmaximum[j]=time1+time2;
            }
            else
                if (j == 3) {
                    num=25;
                    xmaximum[j]=time2+time1+time3;
                }
    }
    setgraph(xmin, xmaximum, xma, yma, &maxx, &maxy, yy, num, curve);
}
else
    if (choice == 3) {
        strcpy(YLABEL, "BRAKING TIME - Seconds");
        strcpy(XLABEL, "BRAKING DISTANCE - Feet");
        xmin=0;
        for (j = 1; j <= curve; j++) {
            if (j == 1) {
                num=3;
                xmaximum[j] = s1_max;
            }
            else
                if (j == 2) {
                    num=26;
                    xmaximum[j] = s2_max+s1_max;
                }
            else
                if (j == 3) {
                    num=27;
                    xmaximum[j] = s3_max+s2_max+s1_max;
                }
        }
    setgraph(xmin, xmaximum, xma, yma, &maxx, &maxy, yy, num, curve);
} }
while ((choice != 1) && (choice != 2) && (choice != 3));
clrscr();
}
else
if (answer == 'N') {
    repeat = FALSE;
    clrscr();
    gotoxy(24, 13); printf("This ends the session on final speed");
    textcolor(RED);
    gotoxy(18, 14); cprintf("Hit the ENTER/RETURN key to return to Speed Menu");
    getchar();
    speed();
}
} while ((answer != 'Y') && (answer != 'N'));
clrscr();
gotoxy(24,13); printf("This ends the session on final speed.");
gotoxy(18,14); printf("Hit the ENTER/RETURN key to return to Main Menu.");
getchar();
mainmenu();

float skid_time(float *p1, float *f1, float *p3, float *s1, float *p5, float *p6)
{
    float max_time, gr;
    float max_friction;
    float yy, dece;
    float fmax_l, dmax;
    int maxx, maxy;
    float xmaximum[4];
    char answer, ch, user_choice[2];
    int choice, repeat;

    repeat = TRUE;
    clrscr();
    textcolor(BLUE);
    gotoxy(8,10); cprintf("The skidding time is the time during which the vehicle produced ");
    gotoxy(8,11); cprintf("the skidmarks. To compute the skidding time, you may use one of ");
    gotoxy(8,12); cprintf("the input parameter sets shown below.");
    gotoxy(8,14); cprintf("* the initial speed, final speed, friction coeff., grade of road");
    gotoxy(8,15); cprintf("* the initial speed, final speed, skidmarks length.");
    gotoxy(8,16); cprintf("* the skidmaks length, friction coefficient and grade of road.");
    press_return();
    clrscr();
do
    {
        textcolor(YELLOW);
        gotoxy(5,10); cprintf("Which parameter set do you want to input?. Type corresponding number 
below");
        textcolor(BLUE);
        gotoxy(10,12); cprintf("1] initial speed, final speed, friction coefficient, grade of road");
        
gotoxy(10,13); cprintf("2] initial speed, final speed, skidmarks length");
gotoxy(10,14); cprintf("3] skidmarks length, friction coefficient, grade of road");
textcolor(YELLOW);
gotoxy(16,16), cprintf("Your choice is (1, 2 or 3) =>");
textcolor(BLUE);
gets(user_choice);
choice=atoi(user_choice);
clrscr();
if (choice= = 1) {
gotoxy(18,12); cprintf("Enter the initial speed of the vehicle (mph) =>");
Vs_max =Get_input(5.0, 200.0);
clrscr();
gotoxy(18,12); cprintf("Enter the final speed of the vehicle (mph) =>");
Ve_max =Get_input(0.0, 200.0);
clrscr();
gotoxy(18,12); cprintf("Enter the friction coefficient of the road =>");
dmax_l =Get_input(0.01, 2.00);
clrscr();
gotoxy(18,12); cprintf("Enter the grade of the road (%) =>");
gr=Get_input(-30, 30);
g=gr/100;
clrscr();
dmax=dmax_l;
time=0.0455*(Vs_max-Ve_max)/(dmax_l +g);
s1_max =s1_max2(Vs_max, Ve_max, dmax_l, g);
dece=(dmax_l +g)*32.2;
clrscr();
gotoxy(15,12); cprintf("The skidding time => %2.2f seconds.", time);
gotoxy(15,13); cprintf("The skidding distance => %2.1f feet", s1_max);
gotoxy(15,14); cprintf("The deceleration => %2.1f feet/sec/sec.", dece);
press_retum();
} else
if (choice= =2) {
gotoxy(18,12); cprintf("Enter the initial speed of the vehicle (mph) =>");
Vs_max =Get_input(5.0, 200.0);
clrscr();
gotoxy(18,12); cprintf("Enter the final speed of the vehicle (mph) =>");
Ve_max =Get_input(0.0, 200.0);
clrscr();
gotoxy(18,12); cprintf("Enter the skidding distance of the vehicle (feet) =>");
s1_max =Get_input(1.0, 2000.0);
clrscr();
gotoxy(18,12); cprintf("Enter the grade of the road (%) =>");
gr=Get_input(-30, 30);
g=gr/100;
time=1.36*s1_max/(Vs_max +Ve_max);
max_friction =fmax_2(Vs_max, Ve_max, s1_max, g);
dece=max_friction*32.2;
dmax_l =max_friction;
clrscr();
The skidding time = %2.2f seconds.
The coefficient of friction = %2.2f,
The deceleration = %2.1f feet/sec/sec.

Enter the skidding distance of the vehicle (feet) =
sl_max = Get_input(1.0, 2000.0);
clrscr();
Enter the coefficient of friction of the road =
dmax_l = Get_input(0.01, 2.00);
clrscr();
Enter the grade of the road (%) =
gr = Get_input(-30, 30);
g = gr/100;
clrscr();
Enter the speed at end of braking (mph) =
Ve_max = Get_input(0.0, 400.0);
Vs_max = sqrt(30*(sl_max*(dmax_l+g))+(Ve_max*Ve_max));
time = 0.0455*(Vs_max-Ve_max)/(dmax_l+g);
dece = (dmax_l+g)*32.2;
clrscr();
The skidding time of the vehicle =

The initial speed of the vehicle = %2.1f mph
The deceleration of the vehicle = %2.1f ft/sec/sec

Do you want to plot the time profile? (Y/N) =
textcolor(YELLOW);
gotoxy(16,7); cprintf("Do you want to plot the time profile? (Y/N) = ");
textcolor(BLUE);
answer = toupper(getche());
getch();
clrscr();
if (answer == 'Y') {
    textcolor(YELLOW);
    do
    {
        gotoxy(18,12); cprintf("Enter the skidding distance of the vehicle (feet) = ");
        sl_max = Get_input(1.0, 2000.0);
        clrscr();
        gotoxy(18,12); cprintf("Enter the coefficient of friction of the road = ");
        dmax_l = Get_input(0.01, 2.00);
        clrscr();
        gotoxy(18,12); cprintf("Enter the grade of the road (%) = ");
        gr = Get_input(-30, 30);
        g = gr/100;
        clrscr();
        gotoxy(18,12); cprintf("Enter the speed at end of braking (mph) = ");
        Ve_max = Get_input(0.0, 400.0);
        Vs_max = sqrt(30*(sl_max*(dmax_l+g))+(Ve_max*Ve_max));
        time = 0.0455*(Vs_max-Ve_max)/(dmax_l+g);
        dece = (dmax_l+g)*32.2;
        clrscr();
        gotoxy(16,12); cprintf("The skidding time of the vehicle =
    ")
        gotoxy(16,13); cprintf("The initial speed of the vehicle = %2.1f mph
    ")
        gotoxy(16,14); cprintf("The deceleration of the vehicle = %2.1f ft/sec/sec
    ");
    }
    while ((choice != 1) && (choice != 2) && (choice != 3));
    *p1 = Vs_max;
    *p5 = Ve_max;
    *s1 = sl_max;
    *f1 = dmax_l;
    *p3 = g;
    *p6 = time;
    /* Drawing graph of skid time versus other parameters. */
    while (repeat)
    {
        do
        {
            textcolor(YELLOW);
            gotoxy(16,7); cprintf("Do you want to plot the time profile? (Y/N) = ");
            textcolor(BLUE);
            answer = toupper(getche());
            getch();
            clrscr();
            if (answer == 'Y') {
                textcolor(YELLOW);
                do
                {
gotoxy(20,9); cprintf("Chose the graph type you want to plot");
textcolor(BLUE);
gotoxy(20,11); printf("1] Distance as a function of Speed");
gotoxy(20,12); printf("2] Time as a function of Distance");
gotoxy(20,13); printf("3] Time as a function of Speed");
textcolor(YELLOW);
gotoxy(20,16); cprintf("Your choice is (1, 2 or 3) = > ");
textcolor(BLUE);
gets(user_choice);
choice = atoi(user_choice);
clrscr();
if (choice == 1) {
    strcpy(YLABEL, "BRAKING DISTANCE - Feet");
    strcpy(XLABEL, "SPEED - Miles per Hour");
    curve=1;
    clrscr();
    xmin=Ve_max;
    num=4;
    yy=s1_max;
    xmaximum[1]=Vs_max;
    setgraph(xmin,xmaximum,xma, yma, &maxx, &maxy, yy, num, curve);
}
else
    if (choice == 2) {
        strcpy(YLABEL, "BRAKING TIME - Seconds");
        strcpy(XLABEL, "BRAKING DISTANCE - Feet");
        xmin=0;
        num=5;
        xmaximum[1]=s1_max;
        yy=time;
        setgraph(xmin,xmaximum,xma, yma, &maxx, &maxy, yy, num, curve);
    }
    else
        if (choice == 3) {
            strcpy(YLABEL, "BRAKING TIME - Seconds");
            strcpy(XLABEL, "SPEED - Miles per Hour");
            curve=1;
            num=12;
            xmin=Ve_max;
            xmaximum[1]=Vs_max;
            yy=time;
            setgraph(xmin,xmaximum,xma, yma, &maxx, &maxy, yy, num, curve);
        }
} while ((choice != 1) &&(choice != 2) &&(choice != 3));
clrscr();
}
else
    if (answer == 'N') {
        clrscr();
        repeat =FALSE;
        textcolor(RED);
float skid_distance(float *p1, float *f1,float *p3,float *s1,float *p5,float *p6)
{
    float t1_max;
    float dece, dmax;
    float max_stop;
    float max_total;
    float pd_max, max_pd;
    float pt_max, ts_max;
    float xmaximum[4];
    float yy, gr;
    float wv;
    float wt;
    int choice;
    int maxx, maxy, repeat;
    char answer, ch, user_choice[2];
    repeat = TRUE;
    textcolor(RED);
    gotoxy(14,10); cprintf("The skidding distance is the distance an automobile");
    gotoxy(14,11); cprintf("travels with locked wheels during braking.");
    textcolor(BLUE);
    gotoxy(14,14); cprintf("To calculate this distance, you may use one of the");
    gotoxy(14,15); cprintf("input parameter sets shown below");
    gotoxy(14,17); cprintf("* initial speed, final speed, friction coeff, grade of road");
    gotoxy(14,18); cprintf("* initial speed, final speed, skidding time, grade of road");
    gotoxy(14,19); cprintf("* skidding time, friction coefficient, final speed, grade of road.");
    press_return();
    clrscr();
    textcolor(YELLOW);
    do
    {
        textcolor(YELLOW);
        gotoxy(5,10); cprintf("Which parameter set do you want to input?. Type corresponding number

below*); textcolor(BLUE);
gotoxy(10,12); cprintf("1] initial speed, final speed, friction coefficient, grade of road");
gotoxy(10,13); cprintf("2] initial speed, final speed, skidding time, grade of road");
gotoxy(10,14); cprintf("3] skidding time, friction coefficient, final speed, grade of road");
textcolor(YELLOW);
gotoxy(16,16), cprintf("Your choice is (1, 2 or 3) = > "); textcolor(BLUE);
gets(user_choice);
choice=atoi(user_choice);
clrscr();
if (choice= = 1) {
    gotoxy(18, 12); cprintf("Enter the initial speed of the vehicle (mph) = > ");
    Vs_max =Get_input(5.0, 200.0);
    clrscr();
    gotoxy(18,12); cprintf("Enter the final speed of the vehicle (mph) = > ");
    Ve_max = Get_input(0.0, 200.0);
    clrscr();
    gotoxy(18,12); cprintf("Enter the friction coefficient of the road = > ");
    dmax_1= Get_input(0.01, 2.00);
    clrscr();
    gotoxy(18,12); cprintf("Enter the grade of the road (%) = > ");
    gr=Get_input(-30, 30);
    g=gr/100;
    clrscr();
    dmax=dmax_1;
do
    textcolor(YELLOW);
    gotoxy(18,12); cprintf("Was the vehicle pulling a brakeless trailer? (Y/N) = > ");
textcolor(BLUE);
    answer = toupper(getche());
    getch();
    if (answer= = 'Y') {
        gotoxy(24,14); cprintf("Enter the weight of the vehicle (pounds) = > ");
        wv =Get_input(100.0,10000.0);
        gotoxy(24,15); cprintf("Enter the weight of the trailer (pounds) = > ");
        wt=Get_input(100.0, 5000.0);
        clrscr();
        s1_max =s1_max1(Vs_max, Ve_max, dmax, g, wv, wt);
        time=0.0455*(Vs_max-Ve_max)/((dmax_1+g)*(wv/(wv+wt)));
        dece=(dmax_1+g)*32.2;
        textcolor(BLUE);
        *s1=s1_max;
        gotoxy(18,13); cprintf("The skidding distance is = %2.1f feet", s1_max);
        gotoxy(18,14); cprintf("The total skidding time is = %2.2f sec.", time);
        gotoxy(18,15); cprintf("The deceleration is = %2.1f ft/sec/sec.", dece);
        getchar();
        press_return();
    }
} else

if (answer == 'N') {
    wv=1.0;
    wt=0.0;
    sl_max =sl_max2(Vs_max, Ve_max, dmax, g);
    time=0.0455*(Vs_max-Ve_max)/((dmax +g)*((wv/(wv+wt));
    dece=(dmax +g)*32.2;
    clrscr();
    gotoxy(18,13); cprintf("The skidding distance is = %2.1f feet", sl_max);
    gotoxy(18,14); cprintf("The skidding time is = %2.2f sec.",time);
    gotoxy(18,15); cprintf("The deceleration is = %2.1f feet/sec/sec.",dece);
    press_return();
}
} while ((answer != 'Y') && (answer != 'N'));
else
    if (choice==2) {
        gotoxy(18,12); cprintf("Enter the grade of the road (%) = > ");
        gr=Get_input(-30, 30);
        g=gr/100;
        clrscr();
        gotoxy(18,12); cprintf("Enter the initial speed of the vehicle (mph) = > ");
        Vs_max = Get_input(5.0, 200.0);
        clrscr();
        gotoxy(18,12); cprintf("Enter the final speed of the vehicle (mph) = > ");
        Ve_max=Get_input(0.0, 200.0);
        clrscr();
        gotoxy(18,12); cprintf("Enter the skidding time of the vehicle (sec.) = > ");
        time =Get_input(0.5, 60.0);
        clrscr();
        s1_max = time*(Vs_max + Ve_max)/1.36;
        dmax _1= (0.0334* (Vs_max*Vs_max)-(Ve_max*Ve_max))/s1_max)-g;
        dece =32.2*(dmax _1+g);
        clrscr();
        gotoxy(15,12); cprintf("The skidding distance = %2.1f feet", s1_max);
        gotoxy(15,13); cprintf("The coefficient of friction = %2.2f", dmax _1);
        gotoxy(15,14); cprintf("The deceleration of the vehicle = %2.1f ft/sec/sec.", dece);
        press_return();
    } else
    if (choice==3) {
        gotoxy(18,12); cprintf("Enter the skidding time of the vehicle (sec) = > ");
        time =Get_input(0.5, 60.0);
        clrscr();
        gotoxy(18,12); cprintf("Enter the coefficient of friction of the road = > ");
        dmax _1=Get_input(0.01, 2.00);
        clrscr();
        gotoxy(18,12); cprintf("Enter the grade of the road (%) = > ");
        gr=Get_input(-30, 30);
        g=gr/100;
        clrscr();
        gotoxy(18,12); cprintf("Enter the speed at end of braking (mph) = > ");
Ve_max=Get_input(0.0, 200.0);
c1rscr();
s1_max = 16.1*time*time*(dmax_l+g) + 1.467*time*Ve_max;
Vs_max = sqrt(30*s1_max*(dmax_l+g) + (Ve_max*Ve_max));
dece=(dmax_l+g)*32.2;
gotoxy(16,12); cprintf("The skidding distance of the vehicle = %2.1f feet.",s1_max);
gotoxy(16,13); cprintf("The initial speed of the vehicle = %2.1f mph ", Vs_max);
gotoxy(16,14); cprintf("The deceleration of the vehicle = %2.1f feet/sec/sec", dece);
presx_return();
}
} while ((choice!=1) && (choice!=2) &&(choice!=3));
*p6=time;
*p1=Vs_max;
*p3=g;
do {

textcolor(YELLOW);
gotoxy(10,12); cprintf("Do you want to calculate the total stopping distance? (Y/N) = > ");
answer=toupper(getche());
if (answer == 'Y') {
textcolor(BLUE);
gotoxy(10,13); cprintf("Total stopping distance is the sum of the perception-reaction");
gotoxy(10,14); cprintf("distance and the braking (skidding) distance of the vehicle. ");
presx_return();
textcolor(YELLOW);
gotoxy(12,12); cprintf("What is the perception-reaction time? = > ");
textcolor(BLUE);
pt_max =Get_input(0.5, 10.0);
max_pd = pd_max(Vs_max, pt_max);
max_total = ts_max(Vs_max, Ve_max, pt_max, dmax_l,g);
gotoxy(12,14); cprintf("The perception_reaction distance = %2.1f feet", max_pd);
gotoxy(12,15); cprintf("The Total Stopping Distance = %2.1f feet", max_total);
presx_return();
}
else
if (answer == 'N') {
clrscr();
}
} while ((answer != 'Y') && (answer != 'N'));
*f1=dmax_l;
*s1=s1_max;
*p5=Ve_max;

/*
Drawing graphs of skid distance versus other parameters. */

while (repeat)
{
do
{
A-27

textcolor(YELLOW);
gotoxy(16,9); cprintf("Do you want to plot the distance profile? (Y/N) = > ");
textcolor(BLUE);
answer = toupper(getche());
getch();
clrscr();
if (answer == 'Y') {
    textcolor(YELLOW);
    do
    {
        gotoxy(20,11); cprintf("Chose the graph-type you want to plot.");
        textcolor(BLUE);
        gotoxy(20,13); cprintf("1] Distance as a function of Speed");
        gotoxy(20,14); cprintf("2] Distance as a function of Time");
        gotoxy(20,15); cprintf("3] Time as a function of Speed");
        textcolor(YELLOW);
        gotoxy(22,18); cprintf("Your choice is (1, 2 or 3) = > ");
        textcolor(BLUE);
        gets(user_choice);
        choice = atoi(user_choice);
        clrscr();
        if (choice == 1) {
            strcpy(YLABEL, "BRAKING DISTANCE - Feet");
            strcpy(XLABEL, "SPEED - Miles per Hour");
            xmin = Ve_max;
            curve = 1;
            num = 8;
            xmaximum[1] = Vs_max;
            yy = sl_max;
            setgraph(xmin, xmaximum, xma, yma, &maxx, &maxy, yy, num, curve);
        }
        else
        if (choice == 2) {
            strcpy(YLABEL, "BRAKING DISTANCE - Feet");
            strcpy(XLABEL, "BRAKING TIME - Seconds");
            curve = 1;
            num = 9;
            xmin = 0.0;
            xmaximum[1] = time;
            yy = s1_max;
            setgraph(xmin, xmaximum, xma, yma, &maxx, &maxy, yy, num, curve);
        }
        else
        if (choice == 3) {
            strcpy(XLABEL, "SPEED - Miles per Hour");
            strcpy(YLABEL, "BRAKING TIME - Seconds");
            curve = 1;
            num = 12;
            xmin = Ve_max;
            xmaximum[1] = Vs_max;
            yy = time;
        }
    }
}
float coeff_friction(float *p1, float *p2, float *p3, float *p4, float *p5, float *p6) {
    float max_friction;
    float xmaximum[4];
    float min, max, delta;
    float wv;
    float wt;
    float yy, gr;
    float t_max, dece;
    char answer, user_choice[2];
    int maxx, maxy, repeat;
    int choice;

    repeat = TRUE;
    textcolor(BLUE);
    gotoxy(13, 10); cprintf("The coefficient of friction is the resulting ratio of the");
    gotoxy(13, 11); cprintf("horizontal force required to overcome resistance in moving");
    gotoxy(13, 12); cprintf("a vehicle or other object along a surface and the weight of");
    gotoxy(13, 13); cprintf("the vehicle or object.");
    gotoxy(13, 15); cprintf("To calculate the coefficient of friction, you may use one of");
    gotoxy(13, 16); cprintf("the input parameter sets shown below.");
    gotoxy(13, 17); cprintf("initial speed, final speed, distance and grade of the road");
    gotoxy(13, 18); cprintf("initial speed, final speed, time and grade of the road");
    gotoxy(13, 19); cprintf("distance, time and grade of the road");
}
press_return();
crsr();
textcolor(YELLOW);
gotoxy(20,13); cprintf("Enter the grade of the road (\%) = > ");
gr= Get_input(-30, 30);
g=g/100;
crsr();
do
{
textcolor(YELLOW);
gotoxy(16,12); cprintf("Did the vehicle stop at the end of skidmarks? (Y/N) = > ");
textcolor(BLUE);
answer = toupper(getche());
if (answer == 'Y') {
    Ve_max=0;
} else
    if (answer == 'N') {
        textcolor(BLUE);
gotoxy(16,14); printf("Enter the final speed of the vehicle (mph) = > ");
        Ve_max = Get_input(0.0, 200.0);
    }
} while ((answer != 'Y') && (answer != 'N'));
crsr();
textcolor(YELLOW);
do
{
gotoxy(16,13); cprintf("Do you know the initial speed of the vehicle? (Y/N) = > ");
answer=toupper(getche());
if (answer== 'Y') {
    gotoxy(16,14); cprintf("Enter the initial speed of the vehicle (mph) = > ");
    Vs_max =Get_input(5.0, 200.0);
crsr();
do
{
textcolor(YELLOW);
gotoxy(18,12); cprintf("Do you know the skidding distance? (Y/N) = > ");
textcolor(BLUE);
answer= toupper(getche());
if (answer == 'Y') {
    gotoxy(18,13); printf("Enter the skidding distance (feet) = > ");
    s1_max =Get_input(1.0, 2000.0);
crsr();
do
{
textcolor(YELLOW);
gotoxy(18,12); cprintf("Was the vehicle pulling a brakeless trailer? (Y/N) = > ");
textcolor(BLUE);
answer = toupper(getche());
if (answer == 'Y') {
gotoxy(24,14); printf("Enter the vehicle's weight (pounds) = > ");
    wv = Get_input(100.0, 10000.0);
gotoxy(24,15); printf("Enter the trailer's weight (pounds) = > ");
    wt = Get_input(100.0, 5000.0);
    clrscr();
    max_friction = fmax_1(Vs_max, Ve_max, s1_max, g, wv, wt);
    t_max = 1.36*s1_max*(wv/(wv+wt))/(Vs_max + Ve_max);
    dece = max_friction*32.2;
    gotoxy(18,13); cprintf("The coefficient of friction is %2.2f", max_friction);
    gotoxy(18,14); cprintf("The total skidding time is %2.2f sec.", t_max);
    gotoxy(18,15); cprintf("The deceleration is %2.1f ft/sec/sec.", dece);
    press_return();
}
else
    if (answer = = 'N') {
        clrscr();
        wv = 1.0;
        wt = 0.0;
        max_friction = fmax_2(Vs_max, Ve_max, s1_max, g);
        t_max = 0.0455*(Vs_max-Ve_max)/max_friction;
        dece = max_friction*32.2;
        gotoxy(18,13); cprintf("The coefficient of friction is %2.2f", max_friction);
        gotoxy(18,14); cprintf("The total skidding time is %2.2f sec.", t_max);
        gotoxy(18,15); cprintf("The deceleration is %2.1f ft/sec/sec.", dece);
        press_return();
    }
} while ((answer != 'Y') && (answer != 'N'));
}
else
    if (answer = = 'N') {
        do

        gotoxy(20,13); printf("Do you know the skidding time? (Y/N) = > ");
        answer = toupper(getche());
        if (answer = = 'Y') {
            gotoxy(20,15); printf("Enter the skidding time = > ");
            t_max = Get_input(0.5, 60.0);
            clrscr();
            max_friction = (Vs_max-Ve_max)/(22*t_max);
            s1_max = (Vs_max+Ve_max)*t_max/1.36;
            dece = (max_friction + g)*32.2;
            gotoxy(18,13); cprintf("The coefficient of friction is %2.2f", max_friction);
            gotoxy(18,14); cprintf("The total skidding distance is %2.1f feet.", s1_max);
gotoxy(18,15); cprintf("The deceleration is %2.1f ft/sec/sec.", dece);
press_return();
}
else
if (answer == 'N') {
    clrscr();
textcolor(RED);
gotoxy(16,12); cprintf("You should have either the skidding distance");
gotoxy(16,13); cprintf("or skidding time to find the friction
coefficient");
gotoxy(16,14); printf(" Hit the ENTER/RETURN key to return to Main
menu.");
getchar();
getchar();
clrscr();
mainmenu();
}
} while ((answer != 'Y') && (answer != 'N'));
}
} while ((answer != 'Y') && (answer != 'N'));
else
if (answer == 'N') {
    clrscr();
do {
    gotoxy(16,11); cprintf("Do you know the skidmarks length and skidding time? (Y/N) => ");
    answer = toupper(getche());
textcolor(BLUE);
    if (answer == 'Y') {
        gotoxy(22,13); cprintf("Enter the skidmarks length (feet) => ");
        sl_max = Get_input(1.0, 2000.0);
        gotoxy(22,14); cprintf("Enter the skidding time (seconds) => ");
        t_max = Get_input(0.5, 60.0);
        Vs_max = (1.36*sl_max/t_max)-Ve_max;
        max_friction = (0.0455*(Vs_max-Ve_max)/(t_max)+g;
        dece = max_friction*32.2;
        clrscr();
textcolor(BLUE);
gotoxy(18,14); cprintf("Coefficient of friction = %2.2f", max_friction);
gotoxy(18,15); cprintf("The sliding speed is %2.1f mph", Vs_max);
gotoxy(18,16); cprintf("The deceleration is %2.1f ft/sec/sec.", dece);
getchar();
press_return();
}
else
if (answer == 'N') {
    clrscr();
textcolor(RED);
gotoxy(18,13); cprintf("Sorry, you cannot find the friction coefficient");
gotoxy(18,14); printf("Hit the RETURN/ENTER key to return to Main Menu.");
getchar();
clrscr();
mainmenu();
}
} while ((answer != 'Y') && (answer != 'N'));
}
} while ((answer != 'Y') && (answer != 'N'));
dmax_l = max_friction;
time = t_max;
*p2 = dmax_l;
*p1 = Vs_max;
*p3 = g;
*p4 = sl_max;
*p5 = Ve_max;
*p6 = time;

/* Drawing graphs of friction coefficient versus other parameters. */

while(repeat)
{
do
{
textcolor(YELLOW);
gotoxy(10,12); cprintf("Do you want to plot the speed, distance and time profiles? (Y/N) = > ");
textcolor(BLUE);
answer = toupper(getche());
getch();
clrscr();
if (answer == 'Y') {

textcolor(YELLOW);
do
{

gotoxy(18,12); cprintf("Chose from below what graph you want to plot");
textcolor(BLUE);
gotoxy(18,14); printf("1] Time as a function of Speed *");
gotoxy(18,15); printf("2] Time as a function of Distance");
gotoxy(18,16); printf("3] Speed as a function of Distance");
textcolor(YELLOW);
gotoxy(22,19); printf("Your choice is (1, 2 or 3) = > ");
textcolor(BLUE);
gets(user_choice);
choice = atoi(user_choice);
clrscr();
if (choice == 1) {
strcpy(XLABEL, "SPEED - Miles per Hour");
strcpy(YLABEL, "BRAKING TIME - Seconds");
num = 12;
curve = 1;
xmin = Ve_max;
xmaximum[1] = Vs_max;
yy = t_max;
setgraph(xmin, xmaximum[1], xma, yma, &maxx, &maxy, yy, num, curve);
}
else
if (choice == 2) {
strcpy(XLABEL, "BRAKING DISTANCE - Feet");
strcpy(YLABEL, "BRAKING TIME - Seconds");
curve = 1;
um = 5;
xmin = 0;
xmaximum[1] = s1_max;
yy = t_max;
setgraph(xmin, xmaximum[1], xma, yma, &maxx, &maxy, yy, num, curve);
}
else
if (choice == 3) {
strcpy(YLABEL, "SPEED - Miles per Hour");
strcpy(XLABEL, "BRAKING DISTANCE - Feet");
xmin = 0;
curve = 1;
um = 1;
xmaximum[1] = s1_max;
yy = Vs_max;
setgraph(xmin, xmaximum[1], xma, yma, &maxx, &maxy, yy, num, curve);
}
} while ((choice != 1) && (choice != 2) && (choice != 3) && (choice != 4));
clrcr();
}
else
if (answer == 'N') {
repeat = FALSE;
clrcr();
textcolor(RED);
gotoxy(20, 12); cprintf("This ends the session on friction coefficient");
gotoxy(20, 13); cprintf("Hit the ENTER/RETURN key to return to Main menu");
getchar();
clrcr();
mainmenu();
}
} while ((answer != 'Y') && (answer != 'N'));
} /* End of repeat */
getchar();
textcolor(RED);
gotoxy(20, 12); cprintf("This ends the session on friction coefficient");
gotoxy(20, 13); printf("Hit the ENTER/RETURN key to return to Main menu");
getchar();
mainmenu();
float Get_input(float lower, float higher)
{
    char temp[80];
    float r;
    int flag = 1;

    while (flag == 1)
    {
        gets(temp);
        if (sscanf(temp, "%f", &r) == 1) {

            if ((r >= lower) && (r <= higher)) flag = 0;
            else printf("Input is out of range -- Try again => ");
        }
        else printf("Invalid input -- Try again => ");
    }
    return (r);
}
/* USA2.C */

#include <stdio.h>
#include <conio.h>
#include <graphics.h>
#include <math.h>
#include <stdlib.h>
#include "usa.h"
#define XTicks 10
#define YTicks 10
#define X(a) x0+a*xc
#define Y(a) y0+a*yc
#define FALSE 0
#define TRUE !FALSE

/* FUNCTIONS */
void press_retum(void);
void road_type();
void road_structure();
void downhill();
void uphill();
void final_speed();
void one_lane(float *p1, float *f1, float *p3, float *s1, float *p5, float *p6, int *p10);
void drag_factorl(float *fl);
float Get_input(float, float);
void drag_factor2(float fmax_1, float fmax_2, float g, float Ve_max);
void combine_drag(float fd_max, float rd_max, float *p5, float *p8, float *p9);
void auto_initialization(void);
void scroll(void);
void setgraph(float xmin, float xmaximum[], float xma, float yma, int *maxx, int *maxy, float yy, int num, int curve);
void plotgraph(float xmin, float xmaximum[], float xma, float yma, float yy, int num, int curve);
float my_function(int num, float x, float y, float dmax_1, float dmax_2, float dmax_3);
float yscale(int num);
float xscale(int num);
void calculator();

/* GLOBAL VARIABLES DEFINED */
extern float Vs_max, s1_max, s2_max, s3_max, time;
extern float Ve_max, dmax_1, dmax_2, dmax_3, g;
extern float fmax_1, fmax_2, fmax_3, rr, Ve1, Ve2;
extern float time1, time2, time3, wv, wt;
extern float xmin, xmax, xma, yma;
extern int num, iabbas, curve;
extern char YLABEL [40];
extern char XLABEL [40];
void setgraph(float xmin, float xmax[], float xma, float yma, int *maxx, int *maxy, float yy, int num, int curve)
{
    int GraphDriver, GraphMode;
    int i, temp, len, x, y, j;
    float x0, y0, xc, yc, size, a, b;
    float temp1, temp2;
    float ymi, xmi;
    char chs[10];
    detectgraph(&GraphDriver, &GraphMode);
    initgraph(&GraphDriver, &GraphMode, "H");
    iabbas = 0;
    setbkcolor(LIGHTGRAY);
    xmi = 0;
    ymi = 0;
    xma = xscale(num);
    yma = yscale(num);
    xc = getmaxx() / (xma - xmi);
    yc = getmaxy() / (yma - ymi);
    x0 = xmi * xc;
    if (x0 < 0)
        x0 *= -1;
    y0 = yma * yc;
    if (y0 < 0)
        y0 *= -1;
    *maxx = getmaxx();
    *maxy = getmaxy();
    setcolor(15);
    line(60, 0, 60, *maxy - 60);
    line(60, *maxy - 60, *maxx, *maxy - 60);
    x = X(xmi);
    y = Y(ymi);
    a = 0;
    gcvt((double)a, 3, chs);
    len = strlen(chs);
    outtextxy(x - len * 4 + 63, *maxy - 50, chs);
    a = 0;
    gcvt((double)a, 3, chs);
    len = strlen(chs);
    outtextxy(x - len * 8 + 55, y - 65, chs);
    /* Scaling the X axis */
    size = (xma - xmi) / XTicks;
    for (i = 1; i <= XTicks; i++)
        { temp = (getmaxx()) / (XTicks + 1);
            line(((temp * i) + 60, *maxy - 60,
                (temp * i) + 60, *maxy - 55);
b = size * i;  
x = temp*i;  
gcvt((double)b, 3, chs);  
len = strlen(chs);  
outtextxy(x-len*8 + 65, *maxy - 50, chs);
}

/* Scaling the Y axis */

x = X(xmi);  
y = Y(ymi);  
size = (yma-ymi)/YTicks;  
for (i = 1; i <= YTicks; i++) {
  temp = (getmaxy()-60)/(YTicks);  
  line(55, (temp*i)-35, 60, (temp*i)-35);  
  a = size * i;  
  gcvt((double)a, 3, chs);  
  len = strlen(chs);  
  outtextxy(x-len*8 + 55, y-(temp*i)-70, chs);
}

setcolor(15);  
settextstyle(1, VERT_DIR, 1);  
moveto(15, (getmaxy()-13)-80);  
outtext(YLABEL);  
settextstyle(1, HORIZ_DIR, 1);  
moveto(getmaxx()/3, getmaxy()-25);  
outtext(XLABEL);  
setcolor(14);  
moveto(getmaxx()/3, 15);  
outtext("Press < RETURN > to continue");  
setcolor(15);  
plotgraph(xmin, xmaximum, xma, yma, yy, num, curve);

void one_lane(float *p1, float *f1, float *p3, float *s1, float *p5, float *p6, int *p10) {

  float dece;  
  float temp1, temp2;  
  float x;  
  int choice, surface;  
  char answer;  
  char user_choice[2];

textcolor(BLUE);  
road_structure();  

textcolor(YELLOW); 
gotoxy(18,13); cprintf("Enter the skidmarks length (feet) = > ");
sl_max = Get_input(1.0, 2000.0);
c1rscr();
do
{
    textcolor(YELLOW);
    gotoxy(18,12); cprintf("Were all wheels locked during braking? (Y/N) = > ");
textcolor(BLUE);
an sorway = toupper(getche());
getch(); 
c1rscr();
if(answer == 'Y') {
    gotoxy(16,12); cprintf("Road friction coefficient is equal to drag factor");
textcolor(YELLOW);
    gotoxy(20,13); cprintf("Enter the road friction coefficient = > ");
fmax_1=Get_input(0.01, 2.00);
}
else
    if (answer == 'N') {
        drag_factor1(&fmax_1);
    }
} while ((answer != 'Y') && (answer != 'N'));
c1rscr();
dmax_1=fmax_1;
do
{
    textcolor(YELLOW);
    gotoxy(15,12); cprintf("Vehicle state at the end of the skidmarks? Chose number below.").
textcolor(BLUE);
    gotoxy(26,14); cprintf(" 1] Vehicle carne to a complete stop. ");
    gotoxy(26,15); cprintf("2] Vehicle did not stop completely. ");
textcolor(YELLOW);
    gotoxy(26,17); cprintf("Your choice is (lor 2) = > ");
textcolor(BLUE);
    gets(user_choice);
    choice =atoi(user_choice);
c1rscr();
    if (choice == 1) {
        Vel=0.0;
    }
else
    if (choice == 2) {
        gotoxy(15,13); printf("Enter the final speed of the vehicle (km/hr) = > ");
        Vel1=Get_input(0.0, 200.0);
    }
} while ((choice != 1) && (choice != 2)); 
*f1=dmax_1;
c1rscr();
textcolor(YELLOW);
do
{
gotoxy(18,12); cprintf("Was the vehicle pulling a brakeless trailer? (Y/N) = > ");
textcolor(BLUE);
answer = toupper(getche());
getch();
if(answer == 'Y') {
    gotoxy(22,14); cprintf("Enter the vehicle's weight (pounds) = > ");
    wv = Get_input(100.0, 10000.0);
    gotoxy(22,15); cprintf("Enter the trailer's weight (pounds) = > ");
    wt = Get_input(100.0, 5000.0);
    clrscr();
    Vs_max = Vs_max1(Vel, sl_max, dmax_l, g, wv, wt);
    time1 = 0.0455*(Vs_max-Vel)/(dmax_l + g);
    dece=(dmax_l + g)*32.2;
    gotoxy(16,13); cprintf("The initial speed of the vehicle = > %2.1f mph", Vs_max);
    gotoxy(16,14); cprintf("The total skidding time = > %2.2f seconds", time1);
    gotoxy(16,15); cprintf("The rate of slowing or deceleration = > %2.1f ft/sec/sec", dece);
}
else
if (answer == 'N') {
    clrscr();
    wv=1.0;
    wt=0.0;
    Vs_max = Vs_max2(Vel, sl_max, dmax_l, g);
    time1 = 0.0455*(Vs_max-Vel)/(dmax_l + g);
    dece=(dmax_l + g)*32.2;
    gotoxy(14,13); cprintf("The initial speed of the vehicle = > %2.1f mph", Vs_max);
    gotoxy(14,14); cprintf("The total skidding time = > %2.2f seconds", time1);
    gotoxy(14,15); cprintf("The rate of slowing or deceleration = > %2.1f ft/sec/sec", dece);
}
}
while ((answer != 'Y') && (answer != 'N'));
*p1 = Vs_max;
*p3 = g;
*p1 = sl_max;
*p5 = Vel1;
*p6 = time1;
curve=1;
*p10=curve;
void two_lane(float *p1, float *f1, float *f2, float *s1, float *s2, float *p3, float *p5, float *p6, float *p7, int *p10)
{
  float dece;
  int choice;
  char answer, user_choice[2];

  road_structure();    /* Displays a sub menu to show road elevation */
  clrscr();
  do
  {
    textcolor(YELLOW);
    gotoxy(15, 10); cprintf("Vehicle state at the end of the skidmarks? Chose number below.");
    textcolor(BLUE);
    gotoxy(26, 12); cprintf("1] Vehicle came to a complete stop.");
    gotoxy(26, 13); cprintf("2] Vehicle did not stop completely.");
    textcolor(YELLOW);
    gotoxy(26, 15); cprintf("Your choice is either (1 or 2) = > ");
    textcolor(BLUE);
    gets(user_choice);
    choice=atoi(user_choice);
    clrscr();
    if (choice == 1)   
        {   
          Ve_max=0;
        }
    else
    if (choice == 2)   
        
        {   
          textcolor(YELLOW);
          gotoxy(13, 13); printf("Enter the final speed of the vehicle (mph) = > ");
          Ve_max =Get_input(1.0, 200.0);
        }
  } while ((choice != 1) && (choice != 2));
  clrscr();
  textcolor(YELLOW);
  textcolor(BLUE);
  gotoxy(16, 12); cprintf("Enter the friction coefficient of surface 1 = > ");
  fmax_1 = Get_input(0.01, 2.00);
  gotoxy(16, 14); cprintf("Enter the friction coefficient of surface 2 = > ");
  fmax_2 = Get_input(0.01, 2.00);
  clrscr();
  do    /* Loop to repeat response if a wrong entry is made */
  
  {    
    textcolor(YELLOW);
    gotoxy(18, 12); cprintf("Were all wheels locked on both surfaces? (Y/N) = > ");
    textcolor(BLUE);
    answer = toupper(getche());
    getch();
  }
if (answer == 'Y') {
    dmax_1 = fmax_1;
    dmax_2 = fmax_2;
}
else
if (answer == 'N') {
/*
    gotoxy(18,13); cprintf(" You need to calculate resultant drag factor"); */
    drag_factor2(fmax_1, fmax_2, g, Ve_max);
    return;
}
} while ((answer != 'Y') && (answer != 'N'));
clrscr();
textcolor(BLUE);
gotoxy(18,12); cprintf("Enter the skidmarks length on surface_1 (feet) = > ");
s1_max = Get_input(1.0, 2000.0);
gotoxy(18,14); cprintf("Enter the skidmarks length on surface_2 (feet) = > ");
s2_max = Get_input(0.0, 2000.0);
clrscr();
do /* Loop to repeat response if a wrong entry is made */
{
textcolor(YELLOW);
gotoxy(18,12); cprintf("Was the vehicle pulling a brakeless trailer? (Y/N) = > ");
textcolor(BLUE);
answer = toupper(getche());
if (answer == 'Y') {
    gotoxy(22,14); cprintf("Enter the vehicle’s weight (pounds) = > ");
wv = Get_input(100.0, 10000.0);
gotoxy(22,16); cprintf("Enter the trailer’s weight (pounds) = > ");
wt = Get_input(100.0, 5000);
getch();
clrscr();
Vs_max = Vs_max3(Ve_max, s1_max, s2_max, dmax_1, dmax_2, g, wv, wt);
Ve1 = sqrt(Vs_max * Vs_max - 30 * s1_max * (dmax_1 + g));
timel = 0.0455 * (Vs_max - Ve1) / (dmax_1 + g);
time2 = 0.0455 * (Ve1 - Ve_max) / (dmax_2 + g);
time = timel + time2;
dece = 1.467 * (Vs_max - Ve_max) / time;
gotoxy(14,15); cprintf("The initial speed of the vehicle = > %2.1f mph", Vs_max);
gotoxy(14,16); cprintf("The total skidding time = > %2.2f seconds", time);
gotoxy(14,17); cprintf("The rate of slowing or deceleration = > %2.1f ft/sec/sec", dece);
}
else
if (answer == 'N') {
    getch();
wv = 1.0;
wt = 0.0;
Vs_max = Vs_max4(Ve_max, s1_max, s2_max, dmax_1, dmax_2, g);
Ve1 = sqrt(Vs_max * Vs_max - 30 * s1_max * (dmax_1 + g));
timel = 0.0455*(Vs_max-Vel)/(dmax_1 + g);
time2 = 0.0455*(Vel-Ve_max)/(dmax_2 + g);
time = timel + time2;
dece = 1.467*(Vs_max-Ve_max)/time;
c1rscc();
gotoxy(14,15); cprintf("The initial speed of the vehicle => %2.1f mph", Vs_max);
gotoxy(14,16); cprintf("The total skidding time => %2.2f seconds", time);
gotoxy(14,17); cprintf("The rate of slowing or deceleration => %2.1f ft/sec/sec", dece);
}
} while ((answer ! = 'Y') && (answer ! = 'N'));

*fl = dmax_1;
*f2 = dmax_2;
*p3 = g;
*p1 = Vs_max;
*s1 = s1_max;
*s2 = s2_max;
*p5 = Ve_max;
*p6 = time1;
*p7 = time2;
curve = 2;
*p10 = curve;

void three_lane(float *pl, float *f1, float *f2, float *f3, float *s1, float *s2, float *s3, float *p3, float *p5, float *p6, float *p7, float *p8, int *p10)
{
    float dece, dmax;
    int choice;
    char answer, user_choice[2];

    road_structure(); /* Displays s sub menu of road elevation */
    clrscr();
textcolor(YELLOW);
gotoxy(14,11); cprintf("Skidding occurred on three different surfaces");
textcolor(BLUE);
gotoxy(14,13); cprintf("Enter the friction coefficient of surface_1 => ");
fmax_1 = Get_input(0.01, 2.00);
gotoxy(14,15); cprintf("Enter the friction coefficient of surface_2 => ");
fmax_2 = Get_input(0.01, 2.00);
gotoxy(14,17); cprintf("Enter the friction coefficient of surface_3 => ");
fmax_3 = Get_input(0.01, 2.00);
c1rscc();
textcolor(YELLOW);
do
{
    textcolor(YELLOW);
gotoxy(15,12); cprintf("Vehicle state at the end of skidmarks. Chose number below");

gotoxy(26,14); printf(" 1] Vehicle came to a complete stop");
gotoxy(26,15); printf(" 2] Vehicle did not stop completely");
gotoxy(26,17); printf(" Your choice is (1 or 2) = ");
textcolor(YELLOW);
gets(user_choice);
choice = atoi(user_choice);
clrscr();
if (choice = = 1) {
    Ve2 = 0;
}
else
    if (choice = = 2) {
        gotoxy(22, 14); printf("Enter the final speed of the vehicle (mph) = ");
        Ve2 = Get_input(1.0, 200.0);
    }
} while ((choice != 1) && (choice != 2));
clrscr();

do  /* Loop to repeat response if a wrong entry is made */
{
    textcolor(YELLOW);
gotoxy(12,11); cprintf("Were all the wheels locked on all the three surfaces? (Y/N) = ");
textcolor(BLUE);
answer = toupper(getche());
getch();
if (answer = = 'Y') {
    dmax_1 = fmax_1;
    dmax_2 = fmax_2;
    dmax_3 = fmax_3;
}
else
/*
gotoxy(18,13); printf("You need to calculate the combined drag factor."); */
    press_return();
    drag_factor2(fmax_1, fmax_2,g, Ve_max);
    return;
}
} while ((answer != 'Y') && (answer != 'N'));

textcolor(BLUE);
gotoxy(12,13); cprintf("Enter the skidmarks length on surface_1 (feet) = ");
s1_max = Get_input(1.0, 2000.0);
gotoxy(12,15); cprintf("Enter the skidmarks length on surface_2 (feet) = ");
s2_max = Get_input(0.0, 2000.0);
gotoxy(12,17); cprintf("Enter the skidmarks length on surface_3 (feet) = ");
s3_max = Get_input(0.0, 2000.0);
crclscr();
textcolor(YELLOW);
do /* loop to repeat response if a wrong entry is made */
{
    textcolor(YELLOW);
gotoxy(14,12); cprintf("Was the vehicle pulling a brakeless trailer? (Y/N) = > ");
textcolor(BLUE);
    answer = toupper(getche());
    getch();
    if (answer == 'Y') {
        gotoxy(18,14); cprintf("Enter the vehicle’s weight (pounds) = > ");
        wv = Get_input(100.0, 10000.0);
        gotoxy(18,15); cprintf("Enter the trailer’s weight (pounds) = > ");
        wt = Get_input(100.0, 5000);
        clrscr();
        Vs_max = Vs_max7 (Ve2,s1_max,s2_max,s3_max,dmax_1,dmax_2,dmax_3,g,wv,wt); (Ve_l)
        Vel = sqrt(Vs_max*Vs_max-30*s1_max*(dmax_1+g));
        Ve_max = sqrt(Vel*Vel-30*s2_max*(dmax_2+g));
        time1 = 0.0455*(Vs_max-Vel)/(dmax_1+g);
        time2 = 0.0455*(Vel-Ve_max)/(dmax_2+g);
        time3 = 0.0455*(Ve_max-Ve2)/(dmax_3+g);
        time = time1 + time2 + time3;
        dece = 1.467*(Vs_max-Ve2)/time;
        gotoxy(16,13); cprintf("The initial speed of the vehicle => %2.1f mph", Vs_max);
        gotoxy(16,14); cprintf("The total skidding time = > %2.2f seconds",time);
        gotoxy(16,15); cprintf("The rate of slowing or deceleration = > %2.1f ft/sec/see*",dece);
        getchar();
    } else
    if (answer == 'N') {
        clrscr();
        wv=1.0;
        wt=0.0;
        Vs_max = Vs_max8 (Ve2,s1_max,s2_max,s3_max,dmax_1,dmax_2,dmax_3,g);
        Vel = sqrt(Vs_max*Vs_max-30*s1_max*(dmax_1+g));
        Ve_max = sqrt(Vel*Vel-30*s2_max*(dmax_2+g));
        time1 = 0.0455*(Vs_max-Vel)/(dmax_1+g);
        time2 = 0.0455*(Vel-Ve_max)/(dmax_2+g);
        time3 = 0.0455*(Ve_max-Ve2)/(dmax_3+g);
        time = time1 + time2 + time3;
        dece = 1.467*(Vs_max-Ve2)/time;
        gotoxy(16,13); cprintf("The initial speed of the vehicle => %2.1f mph", Vs_max);
        gotoxy(16,14); cprintf("The total skidding time = > %2.2f seconds",time);
        gotoxy(16,15); cprintf("The rate of slowing or deceleration = > %2.1f ft/sec/see*",dece);
    }
} while ((answer != 'Y') && (answer != 'N'));

void drag_factor1(float *fl)
{
    float pr, fd_max, dmax;
    float ps, rd_max, x, y;
    int choice;
    char user_choice[2];

do
{
    textcolor(YELLOW);
    gotoxy(14, 10); cprintf("Chose one the options below to find the total drag.");
    textcolor(BLUE);
    gotoxy(14,12); cprintf("1] To provide a reasonable estimate or use the rolling ");
    gotoxy(14,13); cprintf(" friction (0.01) as the drag on the rolling wheels.");
    gotoxy(14,15); cprintf("2] To provide the percentage of the vehicle weight ");
    gotoxy(14,16); cprintf(" carried by the rolling wheels.");
    textcolor(YELLOW);
    gotoxy(26,18); cprintf("Your option is (1 or 2) = > ");
    gets(user_choice);
    choice=atoi(user_choice);
    clrscr();
    if (choice==1) {
        gotoxy(14,14); cprintf("Enter the drag factor (or estimate) of the rolling wheels. = > ");
        rr=Get_input(0.01, 2.00);
        fd_max = rr;
        clrscr();
        gotoxy(18,14); cprintf("Enter the coefficient of friction of the road = > ");
        rd_max = Get_input(0.01, 2.00);
        clrscr();
        combine_drag(fd_max, rd_max, &dmax, &x, &y);
        gotoxy(15,12); cprintf("The longitudinal position of the center of mass = %2.2f ft",y);
    }
}
gotoxy(15,13); cprintf("Height of the center of mass of the vehicle = %2.2f ft", x);
gotoxy(15,14); cprintf("The resultant drag factor = > %2.2f", dmax);
dmax_1 = dmax;
press_return();
}
if (choice == 2) {
    gotoxy(10,14); cprintf("Enter the percentage of the vehicle weight on rolling wheel(s) = > ");
    pr = Get_input(0.0, 100.0);
    ps = 100 - pr;
    clrscr();
    gotoxy(18,12); printf("Enter the friction coefficient of the road = > ");
    fmax_1 = Get_input(0.01, 2.00);
    dmax_1 = 2*(rr *(pr/100)) + 2*((ps/100) * fmax_1);
}
} while ((choice != 1) && (choice != 2));
*f1 = dmax_1;


void drag_factor2(float fmax_1, float fmax_2, float g, float Ve_max)
{
    float rd_max;
    float fd_max;
    float dmax;
    float wv, wt, x, y;
    float dece;
    char answer, user_choice[2];
    int choice;

    do
    {
        textcolor(LIGHTCYAN);
        gotoxy(12,8); cprintf("Describe the path of the wheels during the sliding motion.");
        gotoxy(12,9); cprintf("Chose number below.");
        textcolor(BLUE);
        gotoxy(12,11); cprintf("1] One front wheel slid on pavement_1 and three on pave_2.");
        gotoxy(12,12); cprintf("2] One rear wheel slid on pavement_1 and three on pave_2.");
        gotoxy(12,13); cprintf("3] Two front wheels slid on pave_1 and two rear on pave_2");
        gotoxy(12,14); cprintf("4] Right front and right rear on pave_1 and left front and");
        gotoxy(12,15); cprintf(" left rear on pave_2.");
        textcolor(YELLOW);
        gotoxy(18,17); cprintf("Your choice is (1, 2, 3 or 4) = > ");
        textcolor(BLUE);
        gets(user_choice);
        choice = atoi(user_choice);
clrscr();
textcolor(YELLOW);
if (choice==1) {
    rd_max = fmax_2;
    fd_max = (fmax_1 + fmax_2)/2;
    if (rd_max != fd_max) {
        gotoxy(16, 12); cprintf("Rear axle drag is not equal to the front axle drag");
        combine_drag(fd_max, rd_max, &dmax,&x,&y);
        textcolor(LIGHTCYAN);
    } else {
        gotoxy(18,12); printf("Rear wheels drag factor equals front wheels drag\n");
        dmax = (fd_max + rd_max)/2;
    }
} else {
    gotoxy(18,11); printf("Front wheels drag is equal to the friction coeff. of pave_2");
    fd_max = fmax_2;
    rd_max = (fmax_1 + fmax_2)/2;
    if (rd_max != fd_max) {
        gotoxy(16, 12); cprintf("Rear axle drag is not equal to the front axle drag");
        combine_drag(fd_max, rd_max, &dmax,&x,&y);
    } else {
        gotoxy(18,12); printf("Rear axle drag factor equals front axle drag");
        dmax = (fd_max + rd_max)/2;
    }
} else {
    gotoxy(18,11); printf("Front axle drag is equal to the friction coeff. of pave_1");
    gotoxy(18,12); printf("Rear axle drag is equal to the friction coeff. of pave_2");
    fd_max = fmax_1;
    rd_max = fmax_2;
    if (rd_max != fd_max) {
        gotoxy(18,13); printf("Rear axle drag is not equal to the front axle drag");
        combine_drag(fd_max, rd_max, &dmax,&x,&y);
    } else {
        gotoxy(18,12); printf("Rear axle drag factor equals to front axle drag");
        dmax = (fd_max + rd_max)/2;
        press_return();
    }
} else {
    gotoxy(20,14); cprintf("Front axle drag equals rear axle drag");
A-48

\[
\begin{align*}
fd_{\text{max}} &= \frac{f_{\text{max,1}} + f_{\text{max,2}}}{2}; \\
r_{\text{d, max}} &= fd_{\text{max}}; \\
d_{\text{max}} &= \frac{f_{\text{max,1}} + f_{\text{max,2}}}{2};
\end{align*}
\]

} while ((choice != 1) && (choice != 2) && (choice != 3) && (choice != 4)); clrscr();
gotoxy(16,12); cprintf("Enter the length of the skidmarks (feet) = > ");
s1_{\text{max}} = Get_{\text{input}}(1.0, 2000.0); clrscr(); do {

\{
textcolor(YELLOW);
gotoxy(18,12); cprintf("Was the vehicle pulling a brakeless trailer? (Y/N) = > ");
textcolor(BLUE); answer = toupper(getche());
if (answer == 'Y') {

gotoxy(22,14); printf("Enter the vehicle's weight (pounds) = > ");
wv = Get_{\text{input}}(100.0, 10000.0);
gotoxy(22,16); printf("Enter the trailer's weight (pounds) = > ");
wt = Get_{\text{input}}(100.0, 5000);
Vs_{\text{max}} = Vs_{\text{max,11}}(Ve_{\text{max}},s1_{\text{max}}, d_{\text{max}},g,wv,wt);
}
else
if (answer == 'N') {

wv=1.0;
wt=0.0;
Vs_{\text{max}} = Vs_{\text{max,12}}(Ve_{\text{max}},s1_{\text{max}},d_{\text{max}},g);
}
} while ((answer != 'Y') && (answer !='N')); clrscr();
textcolor(LIGHTCYAN);
time=(1.36*s1_{\text{max}})/(Vs_{\text{max}}+Ve_{\text{max}});
dece=1.467*(Vs_{\text{max}}-Ve_{\text{max}})/time;
gotoxy(15,12); cprintf("The rear axle drag factor = > \%f", rd_{\text{max}});
gotoxy(15,13); printf("The front axle drag factor = > \%f", fd_{\text{max}});
gotoxy(15,14); printf("The resultant drag factor = > \%f", d_{\text{max}});
gotoxy(15,16); cprintf("The initial speed of the vehicle = > \%2.1f mph", Vs_{\text{max}});
gotoxy(15,17); cprintf("The total skidding time = > \%2.2f seconds",time);
gotoxy(15,18); cprintf("The rate of slowing or deceleration = > \%2.1f ft/sec/sec.", dece); getchar();}
void combine_drag(float fd_max, float rd_max, float *p5, float *p8, float *p9)
{
    /* Function determines the resultant drag factor */
    float sl_max;
    float L;
    float R;
    float W;
    float Wr;
    float Wh;
    float h;
    float x;
    float y;
    float dmax;
    char answer;

    textcolor(LIGHTCYAN);
    gotoxy(14,8); cprintf("To find the combine or resultant drag factor,");
    gotoxy(14,9); cprintf("you need to have values of the following parameters.");
    do
    {
        textcolor(BLUE);
        gotoxy(14,11); cprintf("Height of center of mass of the vehicle");
        gotoxy(14,12); cprintf("Length of the vehicle's wheelbase");
        gotoxy(14,13); cprintf("Weight on the rear axle with vehicle level");
        gotoxy(14,14); cprintf("Weight of the rear axle when the front is hoisted h ft");
        gotoxy(14,15); cprintf("Weight of the vehicle");
        gotoxy(14,16); cprintf("Radius of wheel with tire");
        gotoxy(14,18); cprintf("Can you get the values for all the above information? (Y/N) => ");
        textcolor(BLUE);
        answer = toupper(getche());
        clrscr();
        textcolor(YELLOW);
        if (answer == 'Y')
        {
            gotoxy(13,7); cprintf("What is the length of vehicle's wheelbase (feet) = > ");
            L = Get_input(1.0, 50.0);
            gotoxy(13,9); cprintf("What is the radius of wheel with tires (feet) = > ");
            R = Get_input(0.5, 10.0);
            gotoxy(13,11); cprintf("What is the weight of vehicle (pounds) = > ");
            W = Get_input(100.0, 10000.0);
            gotoxy(13,13); cprintf("Weight on the rear axle with vehicle level (pounds) = > ");
            Wr = Get_input(50.0, 5000);
            gotoxy(13,15); cprintf("Weight of rear axle when the front end is hoisted 'h' ft (pounds) = > ");
            Wh = Get_input(50.0, 5000);
            gotoxy(13,17); cprintf("Height to which an axle is hoisted (feet) = > ");
            h = Get_input(0.5, 10.0);
            y = Wr/W;
        }
    }
}
\[ x = (R/L) + ((W_h - W_r) \cdot (L - h)/(h \cdot W)) \]
\[ d_{\text{max}} = \left( \frac{d_{\text{max}} \cdot (1-y) + r_{\text{dmax}} \cdot y}{1-x \cdot (d_{\text{max}} - r_{\text{dmax}})} \right) \]
\* p5 = d_{\text{max}} \\
\* p8 = x \\
\* p9 = y \\
clrsr();

else

if (answer == 'N') {
    textcolor(LIGHTCYAN);
gotoxy(18,12); cprintf("The information is needed to find the combine drag factor.");
gotoxy(18,13); cprintf("before you can find the initial speed of vehicle");
gotoxy(18,14); cprintf("Press ENTER/RETURN key to return to Speed Menu.");
getchar();
speed();
}

} while ((answer ! = 'Y') && (answer ! = 'N')) ;

void plotgraph(float xmin,float xmaximum[],float xma,float yma,float yy,int num,int curve) {
    float result,x,y,gir,result1;
    float xys,xsp,ysp;
    int x_real,y_real,icase, icounter, iabhas;
    int i, maxx, maxy, j, k, counter,counter1;
    FILE *ioptr1;
    FILE *ioptr2;
    FILE *ioptr3;
    int GraphDriver, GraphMode;
    float nur1,nur2,nur3,eric,nur;
    static int hadi[]; {0, 1, 16, 17};
    static int deen[] = {0, 2, 24, 25};
    static int nuru[] = {0, 3, 26, 27};
    if ((num == 1) || (num == 16) || (num == 17)) {
        ioptr1=fopen("mydat1.dat", "w");
        ioptr2 = fopen("mydat2.dat", "w");
        ioptr3 = fopen("mydat3.dat", "w");
        maxx = getmaxx();
        maxy = getmaxy();
x = xmin;
y = yy;
for (j=1; j <= curve; j++) {
do
    
    ys = (maxy-60)/yma;
x = (maxx-60)/xma;
    }
if (j == 1) {
    result = my_function(hadi[j], x, y, dmax_1, dmax_2, dmax_3);
gir = result;
    fprintf(ioptr1, "\n %f %f", x, gir);
}
else if (j == 2) {
    result = my_function(hadi[j], x, y, dmax_1, dmax_2, dmax_3);
    fprintf(ioptr2, "\n %f %f", x, result);
}
else if (j == 3) {
    result = my_function(hadi[j], x, y, dmax_1, dmax_2, dmax_3);
    result_1 = result;
    fprintf(ioptr3, "\n %f %f", x, result_1);
}
xsp = x*xs;
ysp = result*ys;
x_real = 60 + xsp;
y_real = 419 - ysp;
putpixel(x_real, y_real, 15);
x = x + 0.01;
} while (x < x_maximum[j]);

getch();
closegraph();
restorecrtmode();
fclose(ioptr1);
fclose(ioptr2);
fclose(ioptr3);
detectgraph(&GraphDriver, &GraphMode);
initgraph(&GraphDriver, &GraphMode, "");
ioptr1 = fopen("mydat1.dat", "r");
ioptr2 = fopen("mydat2.dat", "r");
ioptr3 = fopen("mydat3.dat", "r");
textmode(3);
setbkcolor(7);
textcolor(1);
textbackground(7);

if (curve == 1) {
    gotoxy(30, 2); cprintf("friction coeff = %2.2f", dmax_1);
gotoxy(30, 3); cprintf("percent grade = %1.2f", g);
gotoxy(30, 5); cprintf("BRAKING SURFACE 1 *");
gotoxy(30, 6); cprintf("DISTANCE SPEED *");
gotoxy(30, 7); cprintf("(feet) (mph) ");
gotoxy(30, 8); cprintf("-----------------------------\n");
    counter1 = 0;
    icounter = 1;
    while (!feof(ioptr1)) {
        for (j = 1; j <= icounter; j = j + 1)
            fscanf(ioptr1, "%f %f", &eric, &nurl);
cprintf("%33.0f %7.1f\n", eric, nur1);
counter = counter + 5;
icounter = counter * 10;
scroll();
}
}
else if (curve == 2) {
gotoxy(24,2); cprintf("friction coeff. of surface 1 = %.2f", dmax_1);
gotoxy(24,3); cprintf("friction coeff. of surface 2 = %.2f", dmax_2);
gotoxy(24,4); cprintf("percent grade of surfaces = %.2f", g);
gotoxy(20,7); cprintf("BRAKING SURFACE 1 SURFACE 2\n");
gotoxy(20,8); cprintf("DISTANCE SPEED SPEED\n");
gotoxy(20,9); cprintf("(feet) (mph) (mph)\n");
gotoxy(20,10); cprintf("----------------------------------------\n");
counter1 = 0;
icounter = 1;
while(!feof(ioptr1)) {
    for (j = 1; j <= icounter; j = j + 1)
        fscanf(ioptr1, "%f %f", &eric, &nur1);
counter = counter + 5;
icounter = counter * 10;
cprintf("%23.0f %7.1f --\n", eric, nur1);
scroll();
}  
while(!feof(ioptr2)) {
    for (j = 1; j <= icounter; j = j + 1)
        fscanf(ioptr2, "%f %f", &eric, &nur2);
cprintf("%23.0f %7.1
", eric, nur2);
counter = counter + 5;
icounter = counter * 10;
scroll();
}
}
else if (curve == 3) {
gotoxy(20,3); cprintf("friction coeff. of surface 1 = %.2f", dmax_1);
gotoxy(20,4); cprintf("friction coeff. of surface 2 = %.2f", dmax_2);
gotoxy(20,5); cprintf("friction coeff. of surface 3 = %.2f", dmax_3);
gotoxy(20,6); cprintf("percent grade of surfaces = %.2f", g);
gotoxy(10,9); cprintf("BRAKING SURFACE 1 SURFACE 2 SURFACE 3\n");
gotoxy(10,10); cprintf("DISTANCE SPEED SPEED SPEED\n");
gotoxy(10,11); cprintf("(feet) (mph) (mph) (mph)\n");
gotoxy(10,12); cprintf("----------------------------------------\n");
counter1 = 0;
icounter = 1;
while(!feof(ioptr1)) {
    for (j = 1; j <= icounter; j = j + 1)
        fscanf(ioptr1, "%f %f", &eric, &nur1);
cprintf("%13.0f %7.1f --\n", eric, nur1);
counter = counter1 + 5;
icounter = 10 * counter;
scroll();
}
while(!feof(ioptr2)) {
    for (j = 1; j <= icounter; j++) {
        fscanf(ioptr2, "%f %f", &eric, &nur2);
        cprintf("%.13.0f %7.1f
", eric, nur2);
        counter = counter1 + 5;
icounter = 10 * counter;
        scroll();
    }
}
while(!feof(ioptr3)) {
    for (j = 1; j <= icounter; j++)
        fscanf(ioptr3, "%f %f", &eric, &nur3);
    cprintf("%.13.0f
", eric, nur3);
    counter = counter1 + 5;
icounter = 10 * counter;
    scroll();
}
press_return();
closegraph();
restorecrtmode();
fclose(ioptr1);
fclose(ioptr2);
fclose(ioptr3);

} else if ((num == 2) || (num == 24) || (num == 25) || (num == 19)) {
ioptr1 = fopen("mydat1.dat", "w");
maxx = getmaxx();
maxy = getmaxy();
x = xmin;
y = yy;
for (j = 1; j <= curve; j++) {
do
    
        ys = (maxy - 60) / yma;
x = (maxx - 60) / xma;
result = my_function(deen[j], x, y, dmax_1, dmax_2, dmax_3);
        fprintf(ioptr1, "%f %f", x, result);
xsp = x * xs;
ysp = result * ys;
x_real = 60 + xsp;
y_real = 419 - ysp;
putpixel(x_real, y_real, 15);
        x = x + 0.01;
    } while (x < xmaximum[j]);
getch();
closegraph();
restorecrtmode();
fclose(ioptr1);
detectgraph(&GraphDriver, &GraphMode);
initgraph(&GraphDriver, &GraphMode, """);
ioptr1 = fopen("mydat1.dat", "r");
textmode(3);
setbkcolor(7);
textcolor(1);
textbackground(7);
textbackground(7);
gotoxy(22,3); cprintf("friction coeff. of surface 1 = %2.2f", dmax_1);
gotoxy(22,4); cprintf("friction coeff. of surface 2 = %2.2f", dmax_2);
gotoxy(22,5); cprintf("friction coeff. of surface 3 = %2.2f", dmax_3);
gotoxy(22,6); cprintf("percent grade of surfaces = %2.2f", g);
gotoxy(30,8); cprintf("BRAKING BRAKING");
gotoxy(30,9); cprintf(" TIME SPEED");
gotoxy(30,10); cprintf("(secs) (mph) ");
gotoxy(30,11); cprintf("--------------------------n");
counter = iabbas = 0;
counter1 = 0;
icounter = 1;
while(!feof(ioptr1))
    { for (j = 1; j <= icounter; j++)
        fscanf(ioptr1,"%f %f", &eric, &nur1);
        cprintf(" %33.2f %7.1f\n", eric, nur1);
        counter = counter1 + 5;
        icounter = counter*10;
        scroll();
    }
press_return();
closegraph();
restorecrtmode();
fclose(ioptr1);
else
    if ((num == 3) || (num == 26) || (num == 27)) 
        { ioptr1 = fopen("mydat1.dat", "w");
            max = getmaxx();
            maxy = getmaxy();
            x = xmin;
            y = yy;
            for (j = 1; j <= curve; j++) 
                do 
                    { 
                        ys = (maxy-60)/yma;
                        xs = (maxx-60)/xma;
                        result = my_function(nuru[j], x, y, dmax_1, dmax_2, dmax_3);
fprintf(ioptr1, "\n f % f", x, result);
 xsp = x * xs;
 ysp = result * ys;
 x_real = 60 + xsp;
 y_real = 419 - ysp;
 putpixel(x_real, y_real, 15);
 x = x + 0.01;
 }
 while (x < xmaxeter[1]);
 }
 getch();
 closegraph();
 restorecrtmode();
 fclose(ioptr1);
 detectgraph(&GraphDriver, &GraphMode);
 initgraph(&GraphDriver, &GraphMode, "H134");
 ioptr1 = fopen("mydat1.dat", "r");
 textmode(3);
 setbkcolor(7);
 textcolor(1);
 textbackground(7);
 gotoxy(22, 3); cprintf("friction coeff. of surface 1 = %2.2f", dmax_1);
 gotoxy(22, 4); cprintf("friction coeff. of surface 2 = %2.2f", dmax_2);
 gotoxy(22, 5); cprintf("friction coeff. of surface 3 = %2.2f", dmax_3);
 gotoxy(22, 6); cprintf("percent grade of surfaces = %2.2f", g);
 gotoxy(30, 9); cprintf("BRAKING BRAKING");
 gotoxy(30, 10); cprintf("TIME DISTANCE");
 gotoxy(30, 11); cprintf("(secs) (feet)" offenses feet);
 gotoxy(30, 12); cprintf("----------------------\n";
 counter1 = 0;
 counter = iabbas = 0;
 icounter = 1;
 while (!feof(ioptr1)) {
 for (j = 1; j <= icounter; j++)
 fscanf(ioptr1, "% f % f", &eric, &nurl);
 cprintf("%33.2f %7.0f\n", eric, nurl);
 counter = counter1 + 5;
 icounter = 10 * counter;
 scroll();
 }
 press_return();
 closegraph();
 restorecrtmode();
 fclose(ioptr1);
 }
 else if (num == 9) {
 ioptr1 = fopen("mydat1.dat", "w");
 maxx = getmaxx();
 maxy = getmaxy();
 x = xmin;

y = yy;
for (j = 1; j <= curve; j++) {
    do
    {
        ys = (maxy - 60) / yma;
        xs = (maxx - 60) / xma;
        result = my_function(num, x, y, dmax_1, dmax_2, dmax_3);
        fprintf(ioptr1, "\n %f %f", x, result);
        xsp = x * xs;
        ysp = result * ys;
        x = 60 + xsp;
        y = 419 - ysp;
        putpixel(x, y, 15);
        x = x + 0.01;
    } while (x < xmaximum[j]);
}
getch();
closegraph();
restorecrtmode();
fclose(ioptr1);
detectgraph(&GraphDriver, &GraphMode);
initgraph(&GraphDriver, &GraphMode, "fI");
ioptr1 = fopen("mydat1.dat", "r");
textmode(3);
setbkcolor(7);
textcolor(1);
textbackground(7);
gotoxy(30, 3); cprintf("friction coefficient = %2.2f", dmax_1);
gotoxy(30, 4); cprintf("percent grade of road = %2.2f", g);
gotoxy(30, 6); cprintf("BRAKING BRAKING");
gotoxy(30, 7); cprintf("TIME DISTANCE");
gotoxy(30, 8); cprintf("(secs) (feet)");
gotoxy(30, 9); cprintf("----------------------");
counter = iabbas = 0;
counter1 = 0;
icounter = 1;
while (!feof(ioptr1)) {
    for (i = 1; i <= icounter; i++)
        fscanf(ioptr1, "%f %f", &eric, &nurl);
        cprintf(" %3.3f %7.2fn", eric, nurl);
        counter = counter + 5;
icounter = 10 * counter;
        scroll();
}
press_return();
closegraph();
restorecrtmode();
fclose(ioptr1);
}
ioptr1 = fopen("mydat1.dat", "w");
maxx = getmaxx();
maxy = getmaxy();
for (j = 1; j <= curve; j++) {
    x = xmin;
    y = yy;
    do
    {
        ys = (maxy - 60)/yma;
        xs = (maxx - 60)/xma;
        result = my_function(num, x, y, dmax_1, dmax_2, dmax_3);
        fprintf(ioptr1, "\n%f %f", x, result);
        xsp = x*xs;
        ysp = result*ys;
        x_real = 60 + xsp;
        y_real = 419 - ysp;
        putpixel(x_real, y_real, 15);
        x = x + 0.1;
    } while (x < xmax);
}
closegraph();
restorecrtmode();
fclose(ioptr1);
detectgraph(&GraphDriver, &GraphMode);
initgraph(&GraphDriver, &GraphMode, "H");
ioptr1 = fopen("mydat1.dat", "r");
textmode(3);
setbkcolor(7);
textcolor(1);
textbackground(7);
if ((num == 4) || (num == 8)) {
    gotoxy(30,3); cprintf("friction coefficient = %2.2f", dmax_1);
    gotoxy(30,4); cprintf("percent grade of road = %2.2f", g);
    gotoxy(30,7); cprintf(" END BRAKING");
    gotoxy(30,8); cprintf("SPEED DISTANCE");
    gotoxy(30,9); cprintf("(mph) (feet)");
    gotoxy(30,10); cprintf("------------------------
    ");
}
else if (num == 12) {
    gotoxy(30,2); cprintf("friction coefficient = %2.2f", dmax_1);
    gotoxy(30,3); cprintf("percent grade of road = %2.2f", g);
    gotoxy(30,4); cprintf("initial speed = %2.1f mph", Vs_max);
    gotoxy(30,6); cprintf("BRAKING BRAKING");
    gotoxy(30,7); cprintf("SPEED TIME");
    gotoxy(30,8); cprintf("(mph) (secs)");
    gotoxy(30,9); cprintf("------------------------
    ");
}
else if (num == 5) {
    gotoxy(30,2); cprintf("friction coefficient = %2.2f", dmax_1);
    gotoxy(30,3); cprintf("percent grade of road = %2.2f", g);
}
gotoxy(30,4); cprintf("initial speed = %2.1f mph", Vs_max);
gotoxy(30,6); cprintf("BRAKING BRAKING");
gotoxy(30,7); cprintf("DISTANCE TIME");
gotoxy(30,8); cprintf("(feet) (secs)");
gotoxy(30,9); cprintf("------------------------
");

counter= iabhas=0;
counter1 =0;
icounter =1;
while (!feof(ioptr1)) {
    for (j=1; j<=icounter; j++)
        fscanf(ioptr1,"%f %f", &eric,&nur1);
    cprintf(" %3.2f %7.2f\n", eric,nur1);
    counter=counter1 +5;
icounter =10*counter;
    scroll();
}

press_return();
closegraph();
restorecrtmode();
fclose(ioptr1);

float my_function(int num, float x, float y, float dmax_1, float dmax_2, float dmax_3) {

    if (num==1) {
        return(sqrt(Vs_max*Vs_max-(30*x*(wv/(wv+wt))*(dmax_1 +g))));
    }
    else
    if (num==2) {
        return(Vs_max-(22*x*(wv/(wv+wt))*(dmax_1 +g)));
    }
    else
    if (num==3) {
        return(1.467*x*Ve1 +16.1*x*x*(wv/(wv+wt))*(dmax_1 +g));
    }
    else
    if (num==4) {
        return(0.0334* ((Vs_max*Vs_max)-(x*x))/(dmax_1 +g));
    }
    else
    if (num==5) {
        return(sqrt(0.0620* x/ (dmax_1 +g) + 0.00208* Ve_max* Ve_max/ ((dmax_1 +g)

else if (num == 6) {
    return(0.0455*(Vs_max-x)/(dmax_1 + g));
}
else if (num == 7) {
    return(0.249*(sqrt(s1_max/(x + g))));
}
else if (num == 8) {
    return(0.0334*((Vs_max*Vs_max)-(x*x))/((wv/(wv+wt))*(dmax_1 + g)));
}
else if (num == 9) {
    return(1.467*x*Ve_max + 16.1*x*x*(wv/(wv+wt))*(dmax_1 + g));
}
else if (num == 12) {
    return((0.0455*(Vs_max-x))/((wv/(wv+wt))*(dmax_1 + g)));
}
else if (num == 13) {
    return(sqrt(0.0620*x/(dmax_1 + g) + 0.00208*Ve_max*Ve_max/((dmax_1 + g)*(dmax_1 + g)))
          - 0.0455*Ve_max/(dmax_1 + g));
}
else if (num == 15) {
    return(sqrt(Vs_max*Vs_max-(30*x*(wv/(wv+wt))*(dmax_1 + g))));
}
else if (num == 16) {
    return(sqrt(Vs_max*Vs_max- (30* (s1_max* (wv/(wv+wt))*(dmax_1 + g)- s1_max*
                                 (wv/(wv+wt)) * (dmax_2 + g)+ x*(wv/(wv+wt))*(dmax_2 + g))));
}
else if (num == 17) {
    return(sqrt(Vs_max*Vs_max-(30* (s1_max* (wv/(wv+wt))*(dmax_1 + g)- s1_max*(wv/
                                 (wv+wt)))* (dmax_2 + g) + (s1_max+s2_max) *(wv/(wv+wt)) *(dmax_2-dmax_3) + x*
                                 (wv/(wv+wt)) * (dmax_3 + g))));
}
else if (num == 18) {
    return(sqrt(y*y-(30*x*(dmax_1 + g))));
}
else if (num == 19) {
    return(y-(22*x*(dmax_1 + g)));
}
else if (num == 24) {
    return(Vs_max - (22* (time1* (wv/(wv+wt)))* (dmax_1+g) - time1* (wv/(wv+wt)))*
            (dmax_2+g) + x*(wv/(wv+wt))*((dmax_2+g)));
}
else if (num == 25) {
    return(Vs_max - (22* (time1* (wv/(wv+wt)))* (dmax_1+g) - time1* (wv/(wv+wt)))*
            (dmax_2+g) + (time1 + time2) *(wv/(wv+wt)) * (dmax_2-dmax_3) + x* (wv/(wv+wt)) *
            (dmax_3+g)));
}
else if (num == 26) {
    return(1.467*time1*Ve1 + 16.1*time1*time1*(wv/(wv+wt))*(dmax_1+g) +
            1.467*(x-time1)*Ve_max + 4.91*(x-time1)*(x-time1)*(wv/(wv+wt))*(dmax_2+g));
}
else if (num == 27) {
    return(1.467*time1*Ve1 + 16.1*time1*time1*(wv/(wv+wt))*(dmax_1+g) + 1.467* time2*
            Ve_max + 16.1* time2 * time2 * (wv/(wv+wt))*(dmax_2+g) + 1.467*(x-time1-time2)* Ve2 +
            16.1* (x-time1-time2)*(x-time1-time2)*(wv/(wv+wt))*(dmax_3+g));
}

float yscale(num)
{
    (num == 5) || (num == 6) || (num == 7) || (num == 10) || (num == 12) || (num == 13) || (num == 14)) {
        return(10);
    }
else if ((num == 3) || (num == 4) || (num == 8) || (num == 9) || (num == 26) || (num == 27)) {
        return(400);
    }
else if ((num == 1) || (num == 2) || (num == 15) || (num == 16) || (num == 17) || (num == 18) ||
            (num == 19) || (num == 24) || (num == 25)) {
        return(100);
    }
float xscale(int num)
{
    if ((num == 1) || (num == 5) || (num == 13) || (num == 14) || (num == 15) 
        || (num == 18) || (num == 22) || (num == 16) || (num == 17)) {
        return(400);
    }
    else if ((num == 2) || (num == 3) || (num == 9) || (num == 19) || (num == 24) 
             || (num == 25) || (num == 26) || (num == 27)) {
        return(10);
    }
    else if ((num == 4) || (num == 6) || (num == 8) || (num == 11) || (num == 12) 
             || (num == 15) || (num == 10)) {
        return(100);
    }
    else if (num == 7) return(1.0);
}

void calculator()
{
    float vs1, vs2, vs3, vs;
    float f1, f2, f3;
    float s1, s2, s3, s;
    float t1, t2, t3, t;
    float speed;
    float friction, f;
    float distance;
    float time;
    int choice, repeat;
    char user_choice[2], answer;

textcolor(YELLOW);
    gotoxy(24, 11); cprintf("Given any two of the input parameters");
    textcolor(BLUE);
    gotoxy(38, 12); cprintf("\* speed");
    gotoxy(38, 13); cprintf("\* distance");
    gotoxy(38, 14); cprintf("\* friction coefficient");
    gotoxy(38, 15); cprintf("\* time");
    textcolor(YELLOW);
    gotoxy(14, 16); cprintf(" the Calculator can compute the remaining two parameters.");
    press_return();
    textcolor(YELLOW);
    repeat = TRUE;
while(repeat)
{
do
{
crlscr();
gotoxy(18,9); cprintf("Which pair of parameters are known? Chose number below");
textcolor(BlUE);
gotoxy(30,11); cprintf(" 1] speed and distance ");
gotoxy(30,12); cprintf(" 2] speed and friction coefficient ");
gotoxy(30,13); cprintf(" 3] speed and time ");
gotoxy(30,14); cprintf(" 4] distance and friction coefficient ");
gotoxy(30,15); cprintf(" 5] distance and time ");
gotoxy(30,16); cprintf(" 6] friction coefficient and time ");
do
{
textcolor(YELLOW);
gotoxy(22,18); cprintf("Your choice is (1, 2, 3, 4, 5 or 6) = > ");
textcolor(BlUE);
gets(user_choice);
choice = atoi(user_choice);
if (choice == 1) {
crlscr();
textcolor(YELLOW);
gotoxy(24,12); cprintf("Enter the vehicle’s speed (mph) = > ");
textcolor(BlUE);
vs = Get_input(5.0, 200.0);
textcolor(YELLOW);
gotoxy(24,14); cprintf("Enter the skidding distance (feet) = > ");
textcolor(BlUE);
s = Get_input(1.0, 500.0);
friction = f1(vs, s);
time = t1(vs, s);
gotoxy(24,16); cprintf("Friction coefficient = %2.2f", friction);
gotoxy(24,17); cprintf("Skidding Time = %2.2f seconds", time);
getchar();
}
else
if(choice == 2) {
crlscr();
gotoxy(24,12); cprintf("Enter the vehicle’s speed (mph) = > ");
vs = Get_input(5.0, 200.0);
gotoxy(24,14); cprintf("Enter the friction coefficient = > ");
f = Get_input(0.01, 2.00);
distance = s1(vs, f);
time = t2(vs, f);
gotoxy(24,16); cprintf("Skidding distance = %2.2f feet", distance);
gotoxy(24,17); cprintf("Skidding Time = %2.2f seconds", time);
getchar();
}
press_return();
}

else
if (choice == 3) {
    clrscr();
gotoxy(24,12); cprintf("Enter the vehicle's speed (mph) = > ");
    vs = Get_input(5.0, 200.0);
gotoxy(24,14); cprintf("Enter the skidding time (secs) = > ");
    t = Get_input(0.5, 60.0);
distance = s2(vs, t);
    friction = f2(vs, t);
gotoxy(24,16); cprintf("Skidding distance = %2.2f feet", distance);
gotoxy(24,17); cprintf("friction coefficient = %2.2f", friction);
    getchar();
    press_return();
} else
if (choice == 4) {
    clrscr();
gotoxy(24,12); cprintf("Enter skidding distance (feet) = > ");
    s = Get_input(5.0, 500.0);
gotoxy(24,14); cprintf("Enter the friction coefficient = > ");
    f = Get_input(0.01, 1.25);
    speed = vsl(s, f);
    time = t3(s, f);
gotoxy(24,16); cprintf("Vehicle speed = %2.2f mph", speed);
gotoxy(24,17); cprintf("Skidding time = %2.2f seconds", time);
    getchar();
    press_return();
} else
if (choice == 5) {
    clrscr();
gotoxy(24,12); cprintf("Enter the skidding distance (feet) = > ");
    s = Get_input(5.0, 500.0);
gotoxy(24,14); cprintf("Enter the skidding time (secs) = > ");
    t = Get_input(0.5, 60.0);
    speed = vs2(s, t);
    friction = f3(s, t);
gotoxy(24,16); cprintf("Vehicle speed = %2.2f mph", speed);
gotoxy(24,17); cprintf("friction coefficient = %2.2f", friction);
    getchar();
    press_return();
} else
if (choice == 6) {
    clrscr();
gotoxy(24,12); cprintf("Enter the friction coefficient = > ");
    f = Get_input(0.01, 1.25);
gotoxy(24,14); cprintf("Enter the skidding time (secs) = > ");
t = Get_input(0.5, 60.0);
speed = vs3(f, t);
distance = s3(f, t);
gotoxy(24, 16); cprintf("Vehicle speed = %.2f mph", speed);
gotoxy(24, 17); cprintf("Skid distance = %.2f feet", distance);
getchar();
press_return();
}
} while (choice != 1) && (choice != 2) && (choice != 3) && (choice != 4) && (choice != 5) && (choice != 6);
gotoxy(16, 13); cprintf("Do you want to continue with the calculator? (Y/N) = ");
answer = toupper(getch());
clrscr();
if (answer == 'Y') {
  repeat = TRUE;
} else
  if (answer == 'N') {
    repeat = FALSE;
    clrscr();
gotoxy(30, 13); cprintf("This ends the session on Calculator");
textcolor(RED);
gotoxy(24, 14); cprintf("Hit the ENTER/RETURN key to return to Main Menu");
getchar();
press_return();
mainmenu();
}
} while ((answer != 'X') && (answer != 'Y'));
/*/ End of repeat */

void press_return( void )
{
  gotoxy(27, 25);
textcolor(YELLOW);
cprintf("press_RETURN- to continue");
getchar();
textcolor(WHITE);
clrscr();
}
void downhill()
{
    float gr;
    textcolor(1);
    gotoxy(20, 12); cprintf("The grade of the road is negative. ");
    gotoxy(20, 13); printf("Enter the grade of the road (%) = > ");
    gr=Get_input(-30, 30);
    g=gr/100;
    clrscr();
}

void uphill()
{
    float gr;
    textcolor(1);
    gotoxy(20, 12); cprintf("The grade of the road is positive. ");
    gotoxy(20, 13); printf("Enter the grade of the road (%) = > ");
    gr=Get_input(-30, 30);
    g=gr/100;
    clrscr();
}

void auto_initialization(void)
{
    int graph_driver;
    int graph_mode;
    graph_driver =DETECT;
    initgraph(&graph_driver, &graph_mode,**);
}

void scroll()
{
    if (iabbas == 14) {
        iabbas =0;
        press_return();
        printf("\n\n\n");
        setbkcolor(7);
        textcolor(1);
        textbackground(7);
    }
    else
        iabbas++;
}
/* -------------------------------------------*/
/* METRIC1.C */
/*------------------------------------------------------*/
/* */
/* A PC Software Package to Determine Speed, Skidding Time, */
/* Skidding Distance and Drag Factor of Vehicles during braking. */
/* Implemented in Turbo C for MS-DOS V 5.0 */
/* */
/* prepared by A. S. Nuruddin as thesis for the degree of */
/* the Master of Science */
/* */
/* Copyright 1993, H.T. Zwahlen Ph.D.; A.S. Nuruddin, Ohio University */
/* Department of Industrial and Systems Engineering */
/* Ohio University, Athens, Ohio 45701-2979 */
/* November 1993 */
/*----------------------------------------------------*/
/* */
/* The following programs are needed to compile this software package. */
/* */
/* */
/* METRIC1.C METRIC2.C METRIC.PRI */
/* METRIC.H UNIT.BAT */
/* */
/* These programs must be pre-compiled and linked */
/* */
/*----------------------------------------------------*/

#include <stdio.h>
#include <math.h>
#include "metric.h"
#include <graphics.h>
#include <conio.h>
#include <alloc.h>
#include <ctype.h>
#include <string.h>
#include <stdlib.h>

#define X(a) x0 + a*xc
#define Y(a) y0 + a*yc
#define XTicks 10
#define YTicks 10
#define Vmax 200
#define smax 400
#define fmax 2.0
#define tmax 10
#ifndef TRUE
#define FALSE 0
#define TRUE !FALSE
#endif
#define X(a) x0 + a*xc
#define Y(a) y0 + a*yc
#define XTicks 10
#define YTicks 10
#define Vmax 200
#define smax 400
#define fmax 2.0
#define tmax 10
#ifndef TRUE
#define FALSE 0
#define TRUE !FALSE
#endif
/* FUNCTIONS */
void title(void);
void auto_initialization(void);
void mainmenu(void);
void press_return(void);
void speed();
void road_type();
char road_structure();
float Get_input(float, float);
void downhill();
void uphill();
float yscale(int num);
float xscale(int num);
void plotgraph(float xma, float xmax, float yma, float yy, int num, int curve);
void setgraph(float xmin, float xmax, float ymin, float ymax, int *maxx, int *maxy, float yy, int num, int curve);
void initial_speed(float *pl, float *fl, float *f2, float *f3, float *s1, float *s2, float *s3, float *p3, float *p5, float *p6, float *p7, float *p8);
void final_speed();
float skid_time(float *p1, float *p2, float *p3, float *p4, float *p5);
float skid_distance(float *p1, float *p2, float *p3, float *p4, float *p5, float *p6);
float coeff_friction(float *p1, float *p2, float *p3, float *p4, float *p5);

/* Global variables defined */
float Vs_max; float Vi_max;
float Ve_max, time, time1, time2, time3;
float fmax_1, fmax_2, fmax_3;
float dmax_1, dmax_2, dmax_3;
float s1_max, s2_max, s3_max, wv, wt, g = 0.0;
float xmin, xmax, ymin, ymax;
int num; int curve;
char XLABEL[40], YLABEL[40];

main()
{
    mainmenu();
}

void mainmenu(void)
{
    char selection;
    float Vs_max;
    float answer;
    float xmin, xmax;
    float ymin, ymax;
    int choice, num;
do
{
    textcolor(BLUE);
    textbackground(LIGHTGRAY);
    clrscr();
    gotoxy(23,4); cprintf("What parameter do you want to find? ");
    gotoxy(10,5); cprintf("Select one of the following by typing the corresponding letter.");
    textcolor(14);
    textbackground(BLUE);
    gotoxy(18,8); cprintf(" A] Initial or Final speed ");
    gotoxy(18,9); cprintf(" B] Skidding distance ");
    gotoxy(18,10); cprintf(" C] Skidding time ");
    gotoxy(18,11); cprintf(" D] Friction coefficient ");
    gotoxy(18,12); cprintf(" E] Calculator ");
    gotoxy(18,13); cprintf(" Q] Quit ");
    gotoxy(18,14); cprintf(" ");
    gotoxy(18,15); cprintf(" ");
    gotoxy(18,16); cprintf(" ");
    gotoxy(18,17); cprintf(" ");
    textbackground(LIGHTGRAY);
    clrscrQ;
    gotoxy(23,4); cprintf(" What parameter do you want to find? ");
    gotoxy(10,5); cprintf(" Select one of the following by typing the corresponding letter. ");
    textcolor(14);
    textbackground(MAGENTA);
    gotoxy(18,18); cprintf(" Your selection is (A, B, C, D, E or Q) = > ");
    gotoxy(18,19); cprintf(" ");
    gotoxy(62,18); textcolor(BLACK);
    selection = toupper(getche());
    gotoxy(62,18); textcolor(LIGHTGRAY);
    clrscr();
    switch (selection)
    {
        case 'A' : speed();
            break;
        case 'B' : skid_distance(&Vs_max,&fmax_l,&g,&s1_max,&Ve_max,&time);
            break;
        case 'C' : skid_time(&Vs_max,&fmax_l,&g,&s1_max,&Ve_max);
            break;
        case 'D' : coeff_friction(&Vs_max,&fmax_l,&g,&s1_max,&Ve_max);
            break;
        case 'E' : calculator();
            break;
        case 'Q' : exit(0);
    } /* end of switch. */
    clrscr();
} /* end of do */
while ((selection ! = 'A')&&(selection ! = 'B')&&(selection ! = 'C')&&(selection ! = 'D')&&(selection
! = 'E')&&(selection ! = 'Q'));
void speed()
{

    char selection;
    clrscr();

do
{
    textcolor(BLUE);
    gotoxy(21,4); cprintf("Which speed type do you want to find? *");
    gotoxy(10,5); cprintf("Select one of the following by typing the corresponding letter");
    textbackground(BLUE);
    textcolor(14);
    gotoxy(18,8); cprintf("**** SPEED MENU ****");
    gotoxy(18,9); cprintf("****");
    gotoxy(18,10); cprintf("*");
    gotoxy(18,11); cprintf("A] Initial_Speed");
    gotoxy(18,12); cprintf("B] Final_Speed");
    gotoxy(18,13); cprintf("C] Main menu");
    gotoxy(18,14); cprintf("Q] Quit");
    gotoxy(18,15); cprintf("*");
    textbackground(MAGENTA);
    textcolor(YELLOW);
    gotoxy(18,16); cprintf("Your selection is (A, B, C or Q) = > ");
    gotoxy(18,17); cprintf("*");
    gotoxy(58,16);
    textcolor(14);
    selection = toupper(getche());
    textbackground(LIGHTGRAY);
    getch();
    switch (selection)
    {
    case 'A' : initial_speed(&Vs_max, &dmax_1, &dmax_2, &dmax_3, &s1_max, &s2_max, &s3_max, &g, &Ve_max, &time1, &time2, &time3, &curve);
        break;
    case 'B' : final_speed();
        break;
    case 'C' : mainmenu();
        break;
    case 'Q' : exit(0);
        press_return();
        clrscr();
    } /* end of switch */
    /* end of do loop */
}while ((selection !='A') &&(selection !='B') && (selection !='C') &&(selection !='Q'));
}
void road_type()
{
    float fmax_2;
    char selection;
    clrscr();
    do
    {
        textcolor(BLUE);
        gotoxy(16,4); cprintf("How many different surfaces did skidding occur?.");
        gotoxy(10,5); cprintf("Select one of the following by typing the corresponding letter");
        textbackground(BLUE);
        textcolor(14);
        gotoxy(18,8); cprintf("**** ROAD SURFACE ****");
        gotoxy(18,9); cprintf("A] One surface");
        gotoxy(18,10); cprintf("B] Two different surfaces");
        gotoxy(18,11); cprintf("C] Three different surfaces");
        gotoxy(18,12); cprintf("D] Return to Speed menu");
        gotoxy(18,13); cprintf("Your selection is (A, B, C or D) = >");
        gotoxy(18,14); cprintf("Your selection is (A, B, C or D) = >");
        gotoxy(18,15); cprintf("Your selection is (A, B, C or D) = >");
        gotoxy(18,16); cprintf("Your selection is (A, B, C or D) = >");
        gotoxy(18,17); cprintf("Your selection is (A, B, C or D) = >");
        textcolor(14);
        gotoxy(18,18); cprintf("Your selection is (A, B, C or D) = >");
        gotoxy(18,19); cprintf("Your selection is (A, B, C or D) = >");
        gotoxy(18,20); cprintf("Your selection is (A, B, C or D) = >");
        gotoxy(18,21); cprintf("Your selection is (A, B, C or D) = >");
        textbackground(MAGENTA);
        textcolor(14);
        gotoxy(18,22); cprintf("Your selection is (A, B, C or D) = >");
        gotoxy(18,23); cprintf("Your selection is (A, B, C or D) = >");
        gotoxy(18,24); cprintf("Your selection is (A, B, C or D) = >");
        gotoxy(18,25); cprintf("Your selection is (A, B, C or D) = >");
        textcolor(BLACK);
        selection = toupper(getche());
        textbackground(LIGHTGRAY);
        getche();
        clrscr();
        switch (selection)
        {
            case 'A' : one_lane(&Vs_max, &dmax_1, &g,&s1_max, &Ve_max,&timel);
                        break;
            case 'B' : two_lane(&Vs_max, &dmax_1, &dmax_2, &s1_max, &s2_1ax, &g,
                            &Ve_max, &time1,&time2);
                        break;
            case 'C' : three_lane(&Vs_max, &dmax_1, &dmax_2, &dmax_3, &s1_max,
                             &s2_max, &s3_max, &g,&Ve_max,&time1,&time2,&time3);
                        break;
            case 'D' : speed();
                        break;
        } /* end of switch */
    } while ((selection != 'A') && (selection != 'B') && (selection != 'C') && (selection != 'D'));
char road_structure()
{
    char selection;
    clrscr();
    do
    {
        textcolor(BLUE);
        gotoxy(23,4); cprintf("Describe the grade of the road.");
        gotoxy(10,5); cprintf("Select one of the following by typing the corresponding letter");
        textbackground(BLUE);
        textcolor(14);
        gotoxy(18,8); cprintf(" **** GRADE OF ROAD **** ");
        gotoxy(18,9); cprintf("A] Level ");
        gotoxy(18,10); cprintf("B] Downhill ");
        gotoxy(18,11); cprintf("C] Uphill ");
        gotoxy(18,12); cprintf("D] Return to Road Surface ");
        textbackground(MAGENTA);
        textcolor(14);
        gotoxy(18,16); cprintf(" Your selection is (A, B, C or D) = > ");
        gotoxy(18,17); cprintf(" ");
        textcolor(BLACK);
        selection = toupper(getche());
        clrscr();
        switch(selection)
        {
            case 'A' : g = 0.0; break;
            case 'B' : downhill(); break;
            case 'C' : uphill(); break;
            case 'D' : road_type();
        }
    } /* end of switch */
    while ((selection != 'A') && (selection != 'B') && (selection != 'C') && (selection != 'D'));
}
void initial_speed(float *p1,float *f1,float *f2,float *f3,float *s1,float *s2,float *s3,float *p3,float *p5,float *p6,float *p7,float *p8,int *p10)
{
    float Vi_max;
    float temp1,temp2;
    float result, yy;
    int repeat;
    int choice,maxx,maxy, itemp;
    char answer, ch, user_choice[2];
    char user_curve[2];
    float xmaximum[4];
    road_type();
    *p1=Vs_max;
    *f1=dmax_1;
    *f2=dmax_2;
    *f3=dmax_3;
    *p3=g;
    *s1=s1_max;
    *s2=s2_max;
    *s3=s3_max;
    *p5=Ve_max;
    *p6=time1;
    *p7=time2;
    *p8=time3;
    *p10=curve;
    press_return();
    repeat = TRUE;
/* Drawing graphs of initial speed versus other parameters */
while(repeat)
{
    do
    {
        textcolor(YELLOW);
        gotoxy(20,7); cprintf("Do you want to plot the speed profile? (Y/N) = > ");
        textcolor(BLUE);
        answer=toupper(getche());
        getch();
        clrscr();
        if (answer == 'Y') {
            textcolor(YELLOW);
            do
            {
                gotoxy(20,9); cprintf("Chose the type of graph you want to plot");
                textcolor(BLUE);
                gotoxy(20,11); cprintf("1] Speed as a function of distance");
                gotoxy(20,12); cprintf("2] Speed as a function of time");
                gotoxy(20,13); cprintf("3] Time as a function of distance");
                textcolor(BLUE);
gotoxy(23,15); cprintf("Your choice is (1, 2 or 3) = > ");
textcolor(BLUE);
gets(user_choice);
choice = atoi(user_choice);
clrscr();
if (choice == 1) {
    strcpy(YLABEL, "SPEED - Kilometers per hour");
    strcpy(XLABEL, "BRAKING DISTANCE - Meters");
    xmin=0.0;
    yy=Vs_max;
    for (itemp=1; itemp<=curve; itemp++) {
        if (itemp==1) {
            num=1;
            xmaximum[itemp] = s1_max;
        }
        else if (itemp==2) {
            num=16;
            xmaximum[itemp] = s2_max+s1_max;
        }
        else if (itemp==3) {
            num=17;
            xmaximum[itemp] = s3_max+s1_max+s2_max;
        }
    }
    setgraph(xmin, xmaximum, xma,yma,&maxx, &maxy,yy,num,curve);
}
else
    if (choice == 2) {
        strcpy(YLABEL, "SPEED - Kilometers per hour");
        strcpy(XLABEL, "BRAKING TIME - Seconds");
        xmin=0.0;
        for (itemp=1; itemp<=curve; itemp++) {
            if (itemp==1) {
                num=2;
                xmaximum[itemp] = time1;
            }
            else
                if (itemp==2) {
                    num=24;
                    xmaximum[itemp] = time2+time1;
                }
                else
                    if (itemp==3) {
                        num=25;
                        xmaximum[itemp] = time3+time2+time1;
                    }
        }
        setgraph(xmin, xmaximum, xma,yma,&maxx, &maxy,yy,num,curve);
    }
    else
        if (choice == 3) {
            strcpy(XLABEL, "BRAKING TIME - Seconds");
strcpy(YLABEL, "BRAKING DISTANCE - Meters");

xmin=0.0;
for (itemp=1; itemp<=curve; itemp++) {
    if (itemp==1) {
        num=3;
        xmaximum[itemp] = timel;
    } else
    if (itemp==2) {
        num=26;
        xmaximum[itemp] = timel + time2;
    } else
    if (itemp==3) {
        num=27;
        xmaximum[itemp] = timel + time2 + time3;
    }
}
setgraphtxmin, xmaximum, xma, yma, &maxx, &maxy, yy, num, curve);
}
} while ((choice != 1) && (choice != 2) && (choice != 3));
clrscr();
else
if (answer == 'N') {
    repeat = FALSE;
crscrt();
gotoxy(24,13); printf("This ends the session on initial speed");
textcolor(RED);
gotoxy(18,14); cprintf("Hit the ENTER/RETURN key to return to Main Menu");
getchar();
mainmenu();
}
while ((answer != 'Y') && (answer != 'N'));
clrscr();
gotoxy(24,13); printf("This ends the session on initial speed.");
textcolor(RED);
gotoxy(18,14); cprintf("Hit the ENTER/RETURN key to return to Main Menu.");
getchar();
mainmenu();

void final_speed()
{
    char selection, user_choice[2];
    float result, yy, dece;
    int repeat, curve, j;
int choice, maxx, maxy, itemp;
char answer, ch;
char user_curve[2];
float xmaximum[4];

clsr();
road_structure();
textcolor(YELLOW);
gotoxy(20,12); cprintf("Enter the initial speed of the vehicle (km/hr) = > ");
Vs_max = Get_input ( 5.0, 400.0);
clsr();
do
{
textcolor(YELLOW);
gotoxy(14,12); cprintf("How many different surfaces are involved. Chose number below");
textcolor(BLUE);
gotoxy(30, 14); cprintf(" 1] one surface");
gotoxy(30,15); cprintf("2] two surfaces");
gotoxy(30,16); cprintf("3] three surfaces");
textcolor(YELLOW);
gotoxy(26,18); cprintf("Your choice is (1, 2, or 3) = > ");
textcolor(BLUE);
gets(user_choice);
choice = atoi(user_choice);
clsr();
if (choice==1) {
    curve=1;
    gotoxy(20,12); printf("Enter the coefficient of friction of the surface = > ");
    dmax_1 = Get_input(0.01, 2.00);
    gotoxy(20,14); printf("Enter the skidmarks length on the surface (meters) = > ");
    s1_max = Get_input(1.0, 2000.0);
clsr();
    Vel=sqrt((Vs_max*Vs_max-253*s1_max*(dmax_1 + g));
    time1 = 0.0283*(Vs_max-Vel)/(dmax_1 + g);
    dece=(dmax_1 + g)*9.81;
    gotoxy(15,13); cprintf("The final speed of the vehicle = > %2.1f km/hr", Vel);
    gotoxy(15,14); cprintf("The total skidding time = > %2.2f sec.", time1);
    gotoxy(15,15); cprintf("The rate of slowing or deceleration = > %2.1f meters/sec/sec.", dece);
    press_return();
}
else if(choice==2) {
    curve=2;
    gotoxy(20,12); printf("Enter the coefficient of friction of surface_1 = > ");
    dmax_1 = Get_input (0.01, 2.00);
    gotoxy(20,14); printf("Enter the coefficient of friction of surface_2 = > ");
    dmax_2 = Get_input (0.01, 2.00);
clsr();
    gotoxy(20,12); printf("Enter the skidmarks length on surface_1 (meters) = > ");
    s1_max = Get_input(1.0, 2000.0);
    gotoxy(20,14); printf("Enter the skidmarks length on surface_2 (meters) = > ");
    s2_max = Get_input(1.0, 2000.0);
 clrscr();
 Vel1=sqrt(Vs_max*Vs_max-253*s1_max*(dmax_1+g));
 Ve_max=sqrt(Vel1*Ve1-253*s2_max*(dmax_2+g));
 time1=0.0283*(Vs_max-Vel1)/(dmax_1+g);
 time2=0.0283*(Ve1-Ve_max)/(dmax_2+g);
 time=time1+time2;
 dece=0.278*(Vs_max-Ve_max)/time;
 gotoxy(15,13); cprintf("The final speed of the vehicle = > %2.1f km/hr", Ve_max);
 gotoxy(15,14); cprintf("The total skidding time = > %2.2f sec.",time);
 gotoxy(15,15); cprintf("The rate of slowing or deceleration = > %2.1f meters/sec/sec.",dece);
 getchar();
 press_return();
 }
 else if(choice==3) { 
 textcolor(BLUE);
 curve=3;
 gotoxy(20,12); cprintf("Enter the coefficient of friction of surface_1 = > ");
 dmax_1=Get_input(0.01, 2.00);
 gotoxy(20,14); cprintf("Enter the coefficient of friction of surface_2 = > ");
 dmax_2=Get_input(0.01, 2.00);
 gotoxy(20,16); cprintf("Enter the coefficient of friction of surface_3 = > ");
 dmax_3=Get_input(0.01, 2.00);
 clrscr();
 gotoxy(20,12); cprintf("Enter the skidmarks length on surface_1 (meters) = > ");
 s1_max =Get_input(1.0, 2000.0); 
 gotoxy(20,14); cprintf("Enter the skidmarks length on surface_2 (meters) = > ");
 s2_max =Get_input(1.0, 2000.0);
 gotoxy(20,16); cprintf("Enter the skidmarks length on surface_3 (meters) = > ");
 s3_max =Get_input(1.0, 2000.0);
 clrscr();
 Vel1=sqrt(Vs_max*Vs_max-253*s1_max*(dmax_1+g));
 Ve_max=sqrt(Vel1*Ve1-253*s2_max*(dmax_2+g));
 Ve2=sqrt(Ve_max*Ve_max-253*s3_max*(dmax_3+g));
 time1=0.0283*(Vs_max-Vel1)/(dmax_1+g);
 time2=0.0283*(Ve1-Ve_max)/(dmax_2+g);
 time3=0.0283*(Ve_max-Ve2)/(dmax_3+g);
 time=time1+time2+time3;
 dece=0.278*(Vs_max-Ve2)/time;
 gotoxy(15,13); cprintf("The final speed the vehicle = > %2.1f km/hr", Ve_max);
 gotoxy(15,14); cprintf("The total skidding time = > %2.2f sec.",time);
 gotoxy(15,15); cprintf("The rate of slowing or deceleration = > %2.1f meters/sec/sec.",dece);
 getchar();
 press_return();
 }
 while ((choice !=1) && (choice !=2) && (choice !=3));

/* Drawing graphs of final speed versus other parameters */
 repeat =TRUE;
 while(repeat)
A-77

{ }
  do
  {
    textcolor(YELLOW);
    gotoxy(12,8); cprintf("Do you want to plot the speed profile? (Y/N) = > ");
    textcolor(BLUE);
    answer=toupper(getche());
    getch();
    clrscr();
    if ( answer== 'Y' ) 
    {
      textcolor(YELLOW);
      gotoxy(20,10); cprintf("Chose the type of graph you want to plot");
      gotoxy(20,12); printf("1] Speed as a function of distance");
      gotoxy(20,13); printf("2] Speed as a function of time");
      gotoxy(20,14); printf("3] Time as a function of distance");
      gotoxy(23,16); cprintf("Your choice is (1, 2 or 3) = > ");
      textcolor(BLUE);
      gets(user_choice);
      choice=atoi(user_choice);
      clrscr();
      if (choice == 1) 
      {
        strcpy(YLABEL, "SPEED - Kilometers per hour");
        strcpy(XLABEL, "BRAKING DISTANCE - Meters");
        xmin=0.0;
        yy = Vs_max;
        num = 18;
        for (itemp=1; itemp <= curve; itemp++)
        {
          if (itemp == 1) 
          {
            num = 1;
            xmaximum[itemp] = s1_max;
          }
          else if (itemp == 2) 
          {
            num = 16;
            xmaximum[itemp] = s2_max + s1_max;
          }
          else if (itemp == 3) 
          {
            num = 17;
            xmaximum[itemp] = s1_max + s2_max + s3_max;
          }
        }
        setgraph(xmin, xmaximum, xma, yma, &maxx, &maxy, yy, num, curve);
      }
      else
      { 
        strcpy(YLABEL, "SPEED - Kilometers per hour");
        strcpy(XLABEL, "TIME REQUIRED TO STOP - Seconds");
        xmin = 0;
        for (j = 1; j <= curve; j++)
        {
          if (j == 1) 
          {
            num = 2;
          }
        }
      }
    }
  }
}
A-78

\[ x_{\text{maximum}}[j] = \text{time1}; \]

else
    if (j == 2) {
        num = 24;
        x_{\text{maximum}}[j] = \text{time1} + \text{time2};
    }

    else
        if (j == 3) {
            num = 25;
            x_{\text{maximum}}[j] = \text{time2} + \text{time1} + \text{time3};
        }

    }

    \text{setgraph}(\text{xmin, x_{\text{maximum}}, x_{\text{ma}}, y_{\text{max}}, \& \text{maxy}}, \text{yy}, \text{num}, \text{curve});

} else
    if (\text{choice} == 3) {
        \text{strcpy(\text{YLABEL}, "BRAKING TIME - Seconds");}
        \text{strcpy(\text{XLABEL}, "BRAKING DISTANCE - Meters");}
        \text{xmin = 0;}
        \text{for (j = 1; j < \text{curve}; j++) { }
            if (j == 1) {
                num = 3;
                x_{\text{maximum}}[j] = s1_{\text{max}};
            }

            else
                if (j == 2) {
                    num = 26;
                    x_{\text{maximum}}[j] = s2_{\text{max}} + s1_{\text{max}};
                }

                else
                    if (j == 3) {
                        num = 27;
                        x_{\text{maximum}}[j] = s3_{\text{max}} + s2_{\text{max}} + s1_{\text{max}};
                    }

                }

        \text{setgraph}(\text{xmin, x_{\text{maximum}}, x_{\text{ma}}, y_{\text{max}}, \& \text{maxy}}, \text{yy}, \text{num}, \text{curve});

    }

while ((\text{choice} != 1) \&\& (\text{choice} != 2) \&\& (\text{choice} != 3));

clfscr();

else
    if (\text{answer} == 'N') {
        \text{repeat} = \text{FALSE;}
        clfscr();
        gotoxy(24, 13); \text{printf("This ends the session on final speed");}
        textcolor(RED);
        gotoxy(18, 14); \text{cprintf("Hit the ENTER/RETURN key to return to Speed Menu");}
        getchar();
        speed();
    }

while ((\text{answer} != 'Y') \&\& (\text{answer} != 'N'));
float skid_time(float *p1, float *f1, float *p3, float *s1, float *p5, float *p6)
{
    float max_time, gr;
    float max_friction;
    float yy, dece;
    float fmax_1, dmax;
    int maxx, maxy;
    float xmaximum[4];
    char answer, ch, user_choice[2];
    int choice, repeat;

    repeat = TRUE;
    clrscr();
    textcolor(BLUE);
    gotoxy(8,10); cprintf("The skidding time is the time during which the vehicle produced ");
    gotoxy(8,11); cprintf("the skidmarks. To compute the skidding time, you may use one of ");
    gotoxy(8,12); cprintf("the input parameter sets shown below.");
    gotoxy(8,14); cprintf("* the initial speed, final speed, friction coeff., grade of road ");
    gotoxy(8,15); cprintf("* the initial speed, final speed, skidmarks length. ");
    gotoxy(8,16); cprintf("* the skidmaks length, friction coefficient and grade of road.");
    press_return();
    clrscr();
    do
    {
        textcolor(YELLOW);
        gotoxy(5,10); cprintf("Which parameter set do you want to input?. Type corresponding 
number below");
        textcolor(BLUE);
        gotoxy(10,12); cprintf("1] initial speed, final speed, friction coefficient, grade of road");
        gotoxy(10,13); cprintf("2] initial speed, final speed, skidmarks length");
        gotoxy(10,14); cprintf("3] skidmarks length, friction coefficient, grade of road");
        textcolor(YELLOW);
        gotoxy(16,16), cprintf("Your choice is (1, 2 or 3) = > ");
        textcolor(BLUE);
        gets(user_choice);
        choice = atoi(user_choice);
        clrscr();
        if (choice == 1) {
            gotoxy(18,12); cprintf("Enter the initial speed of the vehicle (km/hr) = > ");
            Vs_max = Get_input(5.0, 400.0);
            clrscr();
A-SO

gotoxy(18,12); cprintf("Enter the final speed of the vehicle (km/hr) = > ");
Ve_max = Get_input(0.0, 400.0);
clrscr();
gotoxy(18,12); cprintf("Enter the friction coefficient of the road = > ");
dmax_l = Get_input(0.01, 2.00);
clrscr();
gotoxy(18,12); cprintf("Enter the grade of the road (%) = > ");
gr = Get_input(-30, 30);
g = gr/100;
clrscr();
dmax = dmax_l;
time = 0.0283*(Vs_max-Ve_max)/(dmax_l + g);
s1_max = s1_max2(Vs_max, Ve_max, dmax, g);
dece = (dmax_l + g)*9.81;
clrscr();
gotoxy(15,12); cprintf("The skidding time = > %2.2f seconds. ", time);
gotoxy(15,13); cprintf("The skidding distance = > %2.1f meters", s1_max);
gotoxy(15,14); cprintf("The deceleration = > %2.1f meters/sec/sec.", dece);
press_return();
}
else
if (choice==2) {

gotoxy(18,12); cprintf("Enter the initial speed of the vehicle (km/hr) = > ");
Vs_max = Get_input(5.0, 400.0);
clrscr();
gotoxy(18,12); cprintf("Enter the final speed of the vehicle (km/hr) = > ");
Ve_max = Get_input(0.0, 400.0);
clrscr();
gotoxy(18,12); cprintf("Enter the skidding distance of the vehicle (meters) = > ");
s1_max = Get_input(1.0, 2000.0);
clrscr();
gotoxy(18,12); cprintf("Enter the grade of the road (%) = > ");
gr = Get_input(-30, 30);
g = gr/100;
time = 7.20*s1_max/(Vs_max + Ve_max);
max_friction = fmax_2(Vs_max, Ve_max, s1_max, g);
dece = max_friction*9.81;
dmax_1 = max_friction;
clrscr();
gotoxy(15,12); cprintf("The skidding time = > %2.2f seconds.", time);
gotoxy(15,13); cprintf("The coefficient of friction = > %2.2f", dmax_1);
gotoxy(15,14); cprintf("The deceleration = > %2.1f meters/sec/sec.", dece);
press_return();
}
else
if (choice==3) {

gotoxy(18,12); cprintf("Enter the skidding distance of the vehicle (meters) = > ");
s1_max = Get_input(1.0, 2000.0);
clrscr();
gotoxy(18,12); cprintf("Enter the coefficient of friction of the road = > ");
dmax_1 = Get_input(0.01, 2.00);
clrscr();
gotoxy(18,12); cprintf("Enter the grade of the road (%) = > ");
gr=Get_input(-30, 30);
g=gr/100;
clsclr();
go toxy(18,12); cprintf("Enter the speed at end of braking (km/hr) = > ");
Ve_max=Get_input(0.0, 400.0);
Vs_max=sqrt(253*(s1_max*(dmax_1+g))+(Ve_max*Ve_max));
time=0.0283*(Vs_max-Ve_max)/(dmax_1+g);
decel=(dmax_1+g)*9.81;
clsclr();

/*

Drawing graph of skid time versus other parameters. */
while (repeat)
{
do
{
textcolor(YELLOW);
gotoxy(16,7); cprintf("Do you want to plot the time profile? (Y/N) = > ");
textcolor(BLUE);
answer =toupper(getche());
getch();
clsclr();
if (answer = = 'Y') {
   textcolor(YELLOW);
do
{
gotoxy(20,9); cprintf("Chose the graph type you want to plot ");
textcolor(BLUE);
gotoxy(20,11); printf("1] Distance as a function of Speed ");
gotoxy(20,12); printf("2] Time as a function of Distance ");
gotoxy(20,13); printf("3] Time as a function of Speed ");
textcolor(YELLOW);
gotoxy(20,16); cprintf("Your choice is (1, 2 or 3) = > ");
textcolor(BLUE);
gets(user_choice);
choice = atoi (user_choice);
clsclr();
if (choice = =1) {
   strcpy(YLABEL, "BRAKING DISTANCE - Meters");
}
strcpy(XLABEL, "SPEED - Kilometers per Hour");
curve = 1;
crscr();
xmin = Ve_max;
num = 4;
yy = s1_max;
xmaximum[1] = Vs_max;
setgraph(xmin, xmaximum, xma, yma, &maxx, &maxy, yy, num, curve);
}
else
if (choice == 2) {
strcpy(YLABEL, "BRAKING TIME - Seconds");
strcpy(XLABEL, "BRAKING DISTANCE - Meters");
xmin = 0;
num = 5;
xmaximum[1] = s1_max;
yy = time;
setgraph(xmin, xmaximum, xma, yma, &maxx, &maxy, yy, num, curve);
}
else
if (choice == 3) {
strcpy(YLABEL, "BRAKING TIME - Seconds");
strcpy(XLABEL, "SPEED - Kilometers per Hour");
curve = 1;
num = 12;
xmin = Ve_max;
xmaximum[1] = Vs_max;
yy = time;
setgraph(xmin, xmaximum, xma, yma, &maxx, &maxy, yy, num, curve);
}
} while ((choice != 1) && (choice != 2) && (choice != 3));
crscr();
}
else
if (answer == 'N') {
    crscr();
    repeat = FALSE;
textcolor(RED);
gotoxy(24,13); cprintf("This ends the session on skidding time.");
gotoxy(18,14); printf("Hit ENTER/RETURN key to return to MAIN menu");
getchar();
mainmenu();
}
} while ((answer != 'Y') && (answer != 'N'));
} /* End of repeat */
crscr();
textcolor(RED);
gotoxy(24,13); cprintf("This ends the session on skidding time.");
gotoxy(18,14); printf("Hit ENTER/RETURN key to return to MAIN menu");
getchar();
mainmenu();
float skid_distance(float *pI, float *f1,float *p3,float *s1,float *p5,float *p6)
{
    float t1_max;
    float de, dmax;
    float max_stop;
    float max_stop;
    float max_total;
    float pd_max, max_pd;
    float pt_max, ts_max;
    float xmaximum[4];
    float yy, gr;
    float wv;
    float wt;
    int choice;
    int max_x, max_y, repeat;
    char answer, ch, user_choice[2];

    repeat = TRUE;
    textcolor(RED);
    gotoxy(14,10); cprintf("The skidding distance is the distance an automobile");
    gotoxy(14,11); cprintf("travels with locked wheels during braking.");
    textcolor(BLUE);
    gotoxy(14,14); cprintf("To calculate this distance, you may use one of the");
    gotoxy(14,15); cprintf("input parameter sets shown below");
    gotoxy(14,17); cprintf("* initial speed, final speed, friction coeff, grade of road");
    gotoxy(14,18); cprintf("* initial speed, final speed, skidding time, grade of road");
    gotoxy(14,19); cprintf("* skidding time, friction coefficient, final speed, grade of road.");
    press_return();
    clrscr();
    textcolor(YELLOW);
    do
    {
        textcolor(YELLOW);
        gotoxy(5,10); cprintf("Which parameter set do you want to input?. Type corresponding number below");
        textcolor(BLUE);
        gotoxy(10,12); cprintf("1] initial speed, final speed, friction coefficient, grade of road");
        gotoxy(10,13); cprintf("2] initial speed, final speed, skidding time, grade of road");
        gotoxy(10,14); cprintf("3] skidding time, friction coefficient, final speed, grade of road");
        textcolor(YELLOW);
        gotoxy(16,16), cprintf("Your choice is (1, 2 or 3) = ");
        textcolor(BLUE);
        gets(user_choice);
        choice = atoi(user_choice);
        clrscr();
        if (choice == 1) {
            gotoxy(18,12); cprintf("Enter the initial speed of the vehicle (km/hr) = ");
            Vs_max = Get_input(5.0, 400.0);
            clrscr();
            gotoxy(18,12); cprintf("Enter the final speed of the vehicle (km/hr) = ");
        }
Ve_max = Get_input(0.0, 400.0);
crscr();
gotoxy(18,12); cprintf("Enter the friction coefficient of the road = > ");
dmax_1 = Get_input(0.01, 2.00);
crscr();
gotoxy(18,12); cprintf("Enter the grade of the road (%) = > ");
gr = Get_input(-30, 30);
g = gr/100;
crscr();
dmax = dmax_1;
do
{
    textcolor(YELLOW);
    gotoxy(18,12); cprintf("Was the vehicle pulling a brakeless trailer? (Y/N) = > ");
textcolor(BLUE);
    answer = toupper(getche());
    getch();
    if (answer == 'Y') {
        gotoxy(24,14); cprintf("Enter the weight of the vehicle (newtons) = > ");
        wv = Get_input(100.0,10000.0);
        gotoxy(24,15); cprintf("Enter the weight of the trailer (newtons) = > ");
        wt = Get_input(0.0, 5000.0);
        clrscr();
        s1_max = s1_max1(Ve_max, Ve_max, dmax, g, wv, wt);
        time = 0.0283*(Ve_max-Ve_max)/((dmax_1 + g)*(wv/(wv+wt)));
        dece = (dmax_1 + g)*9.81;
        textcolor(BLUE);
        *s1 = s1_max;
        gotoxy(18,13); cprintf("The skidding distance is %2.1f meters", s1_max);
        gotoxy(18,14); cprintf("The total skidding time is %2.2f sec. ", time);
        gotoxy(18,15); cprintf("The deceleration is %2.1f meters/sec/sec.");
        getchar();
        press_return();
    } else
    if (answer == 'N') {
        wv = 1.0;
        wt = 0.0;
        s1_max = s1_max2(Ve_max, Ve_max, dmax, g);
        time = 0.0283*(Ve_max-Ve_max)/((dmax_1 + g)*(wv/(wv+wt)));
        dece = (dmax_1 + g)*9.81;
        clrscr();
        gotoxy(18,13); cprintf("The skidding distance is %2.1f meters", s1_max);
        gotoxy(18,14); cprintf("The skidding time is %2.2f sec.", time);
        gotoxy(18,15); cprintf("The deceleration is %2.1f meters/sec/sec.");
        press_return();
    }
} while ((answer != 'Y') && (answer != 'N'));
else
if (choice == 2) {
    gotoxy(18,12); cprintf("Enter the grade of the road (\%) = > ");
gr = Get_input(-30, 30);
g = gr/100;
crsr();
gotoxy(18,12); cprintf("Enter the initial speed of the vehicle (km/hr) = > ");
Vs_max = Get_input(5.0, 400.0);
crsr();
gotoxy(18,12); cprintf("Enter the final speed of the vehicle (km/hr) = > ");
Ve_max = Get_input(0.0, 400.0);
crsr();
gotoxy(18,12); cprintf("Enter the skidding time of the vehicle (sec.) = > ");
time = Get_input(0.5, 60.0);
crsr();
s1_max = time*(Vs_max + Ve_max)/7.20;
dmax_1 = (0.00393*(Vs_max*Vs_max)-(Ve_max*Ve_max))/s1_max-g;
dece = 9.81*(dmax_1+g);
crsr();
gotoxy(15,12); cprintf("The skidding distance = %2.1f meters", s1_max);
gotoxy(15,13); cprintf("The initial speed of the vehicle = %2.1f km/hr ", Vs_max);
gotoxy(15,14); cprintf("The deceleration of the vehicle = %2.1f meters/sec/sec.", dece);
press_return();
}
else
    if (choice == 3) {
        gotoxy(18,12); cprintf("Enter the skidding time of the vehicle (sec.) = > ");
time = Get_input(0.5, 60.0);
crsr();
gotoxy(18,12); cprintf("Enter the coefficient of friction of the road = > ");
dmax_1 = Get_input(0.01, 2.00);
crsr();
gotoxy(18,12); cprintf("Enter the grade of the road (\%) = > ");
gr = Get_input(-30, 30);
g = gr/100;
crsr();
gotoxy(18,12); cprintf("Enter the speed at end of braking (km/hr) = > ");
Ve_max = Get_input(0.0, 400.0);
crsr();
s1_max = 4.91*time*time*(dmax_1+g) + 0.278*time*Ve_max;
Vs_max = sqrt((253*s1_max*(dmax_1+g)+(Ve_max*Ve_max)));
dece = (dmax_1+g)*9.81;
gotoxy(16,12); cprintf("The skidding distance of the vehicle = %2.1f meters.", s1_max);
gotoxy(16,13); cprintf("The initial speed of the vehicle = %2.1f km/hr ", Vs_max);
gotoxy(16,14); cprintf("The deceleration of the vehicle = %2.1f meters/sec/sec.", dece);
press_return();
}
{
  textcolor(YELLOW);
gotoxy(10,12); cprintf("Do you want to calculate the total stopping distance? (Y/N) = > ");
answer = toupper(getche());
if (answer == 'Y') {
  textcolor(BLUE);
gotoxy(10,13); cprintf("Total stopping distance is the sum of the perception-reaction ");
gotoxy(10,14); cprintf("distance and the braking (skidding) distance of the vehicle.");
press_return();
textcolor(YELLOW);
gotoxy(12,12); cprintf("What is the perception-reaction time? = > ");
textcolor(BLUE);
pt_max = Get_input(0.5, 10.0);
max_pd = pd_max(Vs_max, pt_max);
max_total = ts_max(Vs_max, Ve_max, pt_max, dmax_1, g);
gotoxy(12,14); cprintf("The perception_reaction distance = %2.1f meters", max_pd);
gotoxy(12,15); cprintf("The Total Stopping Distance = %2.1f meters", max_total);
press_return();
} else
if (answer == 'N') {
  clrscr();
}
} while ((answer != 'Y') && (answer != 'N'));
*f1 = dmax_1;
*s1 = s1_max;
*p5 = Ve_max;

/*
   Drawing graphs of skid distance versus other parameters. */
while (repeat)
{
do
{
  textcolor(YELLOW);
gotoxy(16,9); cprintf("Do to want to plot the distance profile? (Y/N) = > ");
textcolor(BLUE);
answer = toupper(getche());
getch();
clrscr();
if (answer == 'Y') {
  textcolor(YELLOW);
do
{
    gotoxy(20,11); cprintf("Chose the graph-type you want to plot.");
textcolor(BLUE);
gotoxy(20,13); cprintf("1] Distance as a function of Speed");
gotoxy(20,14); cprintf("2] Distance as a function of Time");
gotoxy(20,15); cprintf("3] Time as a function of Speed");
}
textcolor(YELLOW);
gotoxy(22,18); cprintf("Your choice is (1, 2 or 3) = > ");
textcolor(BLUE);
gets(user_choice);
choice = atoi (user_choice);
clrscr();
if (choice == 1) {
    strcpy(YLABEL, "BRAKING DISTANCE - Meters");
    strcpy(XLABEL, "SPEED - Kilometers per Hour");
    xmin = Ve_max;
    curve = 1;
    num = 8;
    xmaximum[1] = Vs_max;
    yy = sl_max;
    setgraph(xmin,xmaximum,xma,yma,&maxx,&maxy,yy,num,curve);
}
else
    if (choice == 2) {
        strcpy(YLABEL, "BRAKING DISTANCE - Meters");
        strcpy(XLABEL, "BRAKING TIME - Seconds");
        curve = 1;
        num = 9;
        xmin = 0.0;
        xmaximum[1] = time;
        yy = sl_max;
        setgraph(xmin,xmaximum,xma,yma,&maxx,&maxy,yy,num,curve);
    }
else
    if (choice == 3) {
        strcpy(XLABEL, "SPEED - Kilometers per Hour");
        strcpy(YLABEL, "BRAKING TIME - Seconds");
        curve = 1;
        num = 12;
        xmin = Ve_max;
        xmaximum[1] = Vs_max;
        yy = time;
        setgraph(xmin,xmaximum,xma,yma,&maxx,&maxy,yy,num,curve);
    }
} while ((choice != 1) && (choice != 2) && (choice != 3));
clrscr();
}
else
if (answer == 'N') {
    clrscr();
    repeat = FALSE;
    textcolor(RED);
gotoxy(18,13); cprintf("This ends the session on Skidding distance.");
gotoxy(18,14); printf("Hit ENTER/RETURN key to return to Main menu");
getchar();
mainmenu();
}
} while ((answer != 'Y') && (answer != 'N'));
float coeff_friction(float *p1, float *p2, float *p3, float *p4, float *p5, float *p6)
{
    float max_friction;
    float xmaximum[4];
    float min, max, delta;
    float wv;
    float wt;
    float yy, gr;
    float t_max, dece;
    char answer, user_choice[2];
    int maxx, maxy, repeat;
    int choice;

    repeat = TRUE;
    textcolor(BLUE);
    gotoxy(13,10); cprintf("The coefficient of friction is the resulting ratio of the ");
    gotoxy(13,11); cprintf("horizontal force required to overcome resistance in moving ");
    gotoxy(13,12); cprintf("a vehicle or other object along a surface and the weight of ");
    gotoxy(13,13); cprintf("the vehicle or object.");
    gotoxy(13,15); cprintf("To calculate the coefficient of friction, you may use one of ");
    gotoxy(13,16); cprintf("the input parameter sets shown below.");
    gotoxy(13,17); cprintf(" * initial speed, final speed, distance and grade of the road ");
    gotoxy(13,18); cprintf(" * initial speed, final speed, time and grade of the road ");
    gotoxy(13,19); cprintf(" * distance, time and grade of the road ");
    press_return();
    clrscr();
    textcolor(YELLOW);
    gotoxy(20,13); cprintf("Enter the grade of the road (%) = > ");
    gr = Get_input(-30, 30);
    g = gr/100;
    clrscr();
    do
    {
        textcolor(YELLOW);
        gotoxy(16,12); cprintf("Did the vehicle stop at the end of skidmarks? (Y/N) = > ");
        textcolor(BLUE);
        answer = toupper(getche());
        if (answer == 'Y') {
            Ve_max = 0;
        }
        else
            Ve_max = 0;
    }
if (answer == 'N') {
    textcolor(BLUE);
    gotoxy(16,14); printf("Enter the final speed of the vehicle (km/hr) = > ");
    Ve_max = Get_input(0.0, 400.0);
}
} while ((answer != 'Y') && (answer != 'N'));
clrscr;

textcolor(YELLOW);
do{
    gotoxy(16,13); cprintf("Do you know the initial speed of the vehicle? (Y/N) = > ");
    answer = toupper(getche());
    if (answer == 'Y') {
        gotoxy(16,14); cprintf("Enter the initial speed of the vehicle (km/hr) = > ");
        Vs_max = Get_input(5.0, 400.0);
        clrscr();
        do{
            textcolor(YELLOW);
            gotoxy(18,12); cprintf("Do you know the skidding distance? (Y/N) = > ");
            textcolor(BLUE);
            answer = toupper(getche());
            if (answer == 'Y') {
                gotoxy(18,13); printf("Enter the skidding distance (meters) = > ");
                s1_max = Get_input(1.0, 2000.0);
                clrscr();
                do{
                    textcolor(YELLOW);
                    gotoxy(18,12); cprintf("Was the vehicle pulling a brakeless trailer? (Y/N) = > ");
                    textcolor(BLUE);
                    answer = toupper(getche());
                    if (answer == 'Y') {
                        gotoxy(24,14); printf("Enter the vehicle's weight (newtons) = > ");
                        wv = Get_input(100.0, 10000.0);
                        gotoxy(24,15); printf("Enter the trailer's weight (newtons) = > ");
                        wt = Get_input(100.0, 5000.0);
                        clrscr();
                        max_friction = fmax_1(Vs_max, Ve_max, s1_max, g, wv, wt);
                        t_max = 7.20*s1_max*(wv/(wv+wt))/(Vs_max+Ve_max);
                        dece = max_friction*9.81;
                        gotoxy(18,13); printf("The coefficient of friction is %2.2f, 
                        max_friction);" max_friction);
                        gotoxy(18,14); printf("The total skidding time is %2.2f sec.,t_max);" sec.,t_max);
                        gotoxy(18,15); printf("The deceleration is %2.1f meters/sec/sec.,dece);
                        press_return();
                    } else
                }
            }
        }
    }
}
} else
if (answer == 'N') {
    clrscr();
    wv=1.0;
    wt=0.0;
    max_friction = fmax_2(Vs_max, Ve_max, sl_max, g);
    t_max=0.0283*(Vs_max-Ve_max)/max_friction;
    dece = max_friction*9.81;
    gotoxy(18,13); cprintf("The coefficient of friction is = %2.2f", max_friction);
    gotoxy(18,14); cprintf("The total skidding time is = %2.2f sec.", t_max);
    gotoxy(18,15); cprintf("The deceleration is = %2.1f meters/sec/sec.", dece);
    press_return();
}
} while ((answer != 'Y') && (answer != 'N'));
else
if (answer == 'N') {
    do
    {
        gotoxy(20,13); printf("Do you know the skidding time? (Y/N) = > ");
        answer=toupper(getche());
        if (answer == 'Y') {
            gotoxy(20,15); cprintf("Enter the skidding time = > ");
            t_max = Get_input(0.5, 60.0);
            clrscr();
            max_friction = (Vs_max-Ve_max)/(35.3*t_max);
            sl_max=(Vs_max+Ve_max)*t_max/7.20;
            dece = (max_friction+g)*9.81;
            gotoxy(18,13); cprintf("The coefficient of friction is = %2.2f", max_friction);
            gotoxy(18,14); cprintf("The total skidding distance is = %2.1f meters.", sl_max);
            gotoxy(18,15); cprintf("The deceleration is = %2.1f meters/sec/sec.", dece);
            press_return();
        }
        else
        if (answer == 'N') {
            clrscr();
            textcolor(RED);
            gotoxy(16,12); cprintf("You should have either the skidding distance");
            gotoxy(16,13); cprintf("or skidding time to find the friction coefficient");
            gotoxy(16,14); cprintf("Hit the ENTER/RETURN key to return to Main menu.");
            getchar();
            getchar();
            clrscr();
            mainmenu();
        }
    }
} else
if (answer == 'N') {
    clrscr();
    textcolor(RED);
    gotoxy(16,12); cprintf("You should have either the skidding distance");
    gotoxy(16,13); cprintf("or skidding time to find the friction coefficient");
    gotoxy(16,14); cprintf("Hit the ENTER/RETURN key to return to Main menu.");
    getchar();
    getchar();
    clrscr();
    mainmenu();
}
```c
} while (!(answer == 'Y') && (answer != 'N'));

} while (!(answer == 'Y') && (answer != 'N'));

} else
if (answer == 'N') {
  clrscr();
  gotoxy(16,11); cprintf("Do you know the skidmarks length and skidding time? (Y/N) = > ");
  answer = toupper(getche());
  textcolor(BLUE);
  if (answer == 'Y') {
    gotoxy(22,13); cprintf("Enter the skidmarks length (meters) = > ");
    s1_max = Get_input(1.0, 2000.0);
    gotoxy(22,14); cprintf("Enter the skidding time (seconds) = > ");
    t_max = Get_input(0.5, 60.0);
    Vs_max = (7.20*s1_max/t_max)-Ve_max;
    max_friction = (0.0283*(Vs_max-Ve_max)/t_max)+g;
    dece = max_friction*9.81;
    clrscr();
    textcolor(BLUE);
    gotoxy(18,14); cprintf("Coefficient of friction = %2.2f", max_friction);
    gotoxy(18,15); cprintf("The sliding speed is = %2.1f km/hr", Vs_max);
    gotoxy(18,16); cprintf("The deceleration is = %2.1f meters/sec/sec.", dece);
    getchar();
    press_return();
  }
else
if (answer == 'N') {
  clrscr();
  textcolor(RED);
  gotoxy(18,13); cprintf("Sorry, you cannot find the friction coefficient");
  gotoxy(18,14); printf("Hit the RETURN/ENTER key to return to Main Menu.");
  getchar();
  clrscr();
  mainmenu();
}
while (!(answer == 'Y') && (answer != 'N'));

} while (!(answer == 'Y') && (answer != 'N'));

dmax_1 = max_friction;
time = t_max;
*p2 = dmax_1;
*p1 = Vs_max;
*p3 = g;
*p4 = s1_max;
*p5 = Ve_max;
*p6 = time;

/* Drawing graphs of friction coefficient versus other parameters. */
while(repeat)
{
    do
    {
        textcolor(YELLOW);
gotoxy(10,12); cprintf("Do you want to plot the speed, distance and time profiles? (Y/N) = > ");
textcolor(BLUE);
answer=toupper(getche());
getch();
clrscr();
if (answer == 'Y') {
    textcolor(YELLOW);
    do
    {
        gotoxy(18,12); cprintf("Chose from below what graph you want to plot");
textcolor(BLUE);
gotoxy(18,14); printf("[1] Time as a function of Speed ");
gotoxy(18,15); printf("[2] Time as a function of Distance");
gotoxy(18,16); printf("[3] Speed as a function of Distance");
textcolor(YELLOW);
gotoxy(22,19); cprintf("Your choice is (1, 2 or 3) = > ");
textcolor(BLUE);
gets(user_choice);
choice = atoi(user_choice);
clrscr();
if (choice == 1) {
    strcpy(XLABEL, "SPEED - Kilometers per Hour");
    strcpy(YLABEL, "BRAKING TIME - Seconds");
    num=12;
    curve=1;
    xmin=Ve_max;
    xmaxinimum[1]=Vs_max;
    yy=t_max;
    setgraph(xmin,xmaximum,xma,yma,&maxx,&maxy,yy,num,curve);
}
else
if (choice == 2) {
    strcpy(XLABEL, "BRAKING DISTANCE - Meters");
    strcpy(YLABEL, "BRAKING TIME - Seconds");
    curve=1;
    num=5;
    xmin=0;
    xmaxinimum[1]=s1_max;
    yy=t_max;
    setgraph(xmin,xmaximum,xma,yma,&maxx,&maxy,yy,num,curve);
}
else
if (choice == 3) {
    strcpy(YLABEL, "SPEED - Kilometers per Hour");
    strcpy(XLABEL, "BRAKING DISTANCE - Meters");
    xmin=0;
}
curve = 1;
num = 1;
xmaximum[1] = s1_max;
yy = Vs_max;
setgraph(xmin, xmax, xma, yma, &maxx, &maxy, yy, num, curve);
} while ((choice != 1) && (choice != 2) && (choice != 3) && (choice != 4));
clrscr();
} else
if (answer = = 'N') {
    repeat = FALSE;
crscr();
textcolor(RED);
gotoxy(20, 12); cprintf("This ends the session on friction coefficient");
gotoxy(20, 13); printf("Hit the ENTER/RETURN key to return to Main menu");
getchar();
crscr();
mainmenu();
}
} while ((answer != 'Y') && (answer != 'N'));
} /* End of repeat */
getchar();
textcolor(RED);
gotoxy(20, 12); cprintf("This ends the session on Coefficient of Friction");
gotoxy(20, 13); printf("Hit the ENTER/RETURN key to return to Main menu");
getchar();
mainmenu();

float Get input(float lower, float higher)
{
    char temp[80];
    float r;
    int flag = 1;

    while (flag == 1)
    {
        gets(temp);
        if (sscanf(temp, "%f", &r) == 1) {
            if ((r >= lower) && (r <= higher)) flag = 0;
            else printf("Input is out of range -- Try again = > ");
        } else
            printf("Invalid input -- Try again = > ");
    }
    return (r);
}
/* METRIC2.C */

#include <stdio.h>
#include <conio.h>
#include <graphics.h>
#include <math.h>
#include <stdlib.h>
#include "metric.h"
#define XTicks 10
#define YTicks 10
#define X(a) xO+a*xc
#define Y(a) yO+a*yc
#define FALSE 0
#define TRUE !FALSE

/* FUNCTIONS */
void press_return(void);
void downhill();
void uphill();
void one_lane(float *p1, float *f1, float *p3, float *s1, float *p5, float *p6, int *p10);
void drag_factor1(float *f1);
float Get_input(float, float);
void drag_factor2(float fmax_1, float fmax_2, float g, float Ve_max);
void combine_drag(float fd_max, float rd_max, float *p5, float *p8, float *p9);
void auto_initialization(void);
void scroll(void);
void setgraph(float xmin, float xmaximum[], float xma, float yma, int *maxx, int *maxy, float yy, int num, int curve);
void plotgraph(float xmin, float xmaximum[], float xma, float yma, float yy, int num, int curve);
float my_function(int num, float x, float y, float dmax_1, float dmax_2, float dmax_3);
float yscale(int Dum);
float xscale(int Dum);
void calculaton();

/* GLOBAL VARIABLES DEFINED */
extern float Vs_max, s1_max, s2_max, s3_max, time;
extern float Ve_max, dmax_1, dmax_2, dmax_3, g;
extern float fmax_1, fmax_2, fmax_3, rr, Ve1, Ve2;
extern float time1, time2, time3, wv, wt;
extern float xmin, xmax, xma, yma;
extern int num, iabbas, curve;
extern char YLABEL [40];
extern char XLABEL [40];
void setgraph(float xmin, float xmax[], float xma, float yma, int *maxx, int *maxy, float yy, int num, int curve)
{
    int GraphDriver, GraphMode;
    int i, temp, len, x, y, j;
    float x0, y0, xc, yc, size, a, b;
    float temp1, temp2;
    float ymi, xmi;
    char chs[10];
    detectgraph(&GraphDriver, &GraphMode);
    initgraph(&GraphDriver, &GraphMode, "") ;

    *maxx = getmaxx();
    *maxy = getmaxy();
    setcolor(15);
    line(60,0,60, *maxy-60);
    line(60, *maxy-60, *maxx, *maxy-60);
    x=X(xmi);
    y=Y(ymi);
    a=0;
    gcvt((double)a,3,chs);
    len=strlen(chs);
    outtextxy(x-len*4+63, *maxy-50, chs);
    a=0;
    gcvt((double)a,3,chs);
    len=strlen(chs);
    outtextxy(x-len*8+55, y-65, chs);

    /*
    Scaling the X axis
    */

    size = (xma-xmi)/XTicks;
    for (i=1; i<=XTicks; i++) {
        temp = (getmaxx())/(XTicks+1);
        line(((temp)*i)+60, *maxy-60,
             (temp)*i)+60, *maxy-55);
void one_lane(float *p1, float *f1, float *p3, float *s1, float *p5, float *p6, int *p10)
{
    float dece;
    float temp1, temp2;
    float x;
    int choice, surface;
    char answer;
    char user_choice[2];

    textcolor(BLUE);
    road_structure();
    textcolor(YELLOW);
gotoxy(18,13); cprintf("Enter the skidmarks length (meters) = > ");
sl_max = Get_input(1.0, 2000.0);
clrscr();
do
{
  textcolor(YELLOW);
gotoxy(18,12); cprintf("Were all wheels locked during braking? (Y/N) = > ");
textcolor(BLUE);
  answer = toupper(getche());
  getch();
clrscr();
  if(answer == 'Y') {
    gotoxy(16,12); cprintf("Road friction coefficient is equal to drag factor");
    textcolor(YELLOW);
    gotoxy(20, 13); cprintf("Enter the road friction coefficient = > ");
    fmax_1=Get_input(0.01, 2.00);
  }
  else
    if (answer == 'N') {
      drag_factor1(&fmax_1);
    }
} while ((answer != 'Y') && (answer!='N'));
clrscr();
dmax_1=fmax_1;
do
{
  textcolor(YELLOW);
  gotoxy(15,12); cprintf("Vehicle state at the end of the skidmarks? Chose number below.");
  textcolor(BLUE);
  gotoxy(26,14); cprintf("1] Vehicle came to a complete stop.");
  gotoxy(26,15); cprintf("2] Vehicle did not stop completely.");
  textcolor(YELLOW);
  gotoxy(26,17); cprintf("Your choice is (1 or 2) = > ");
  textcolor(BLUE);
  gets(user_choice);
  choice =atoi(user_choice);
  clrscr();
  if (choice == 1) {
    Vel=0.0;
  }
  else
    if (choice == 2) {
      gotoxy(15,13); printf("Enter the final speed of the vehicle (km/hr) = > ");
      Vel=Get_input(0.0, 200.0);
    }
} while ((choice !=1) && (choice !=2));
*fl=dmax_1;
clrscr();
textcolor(YELLOW);
do
{
gotoxy(18,12); cprintf("Was the vehicle pulling a brakeless trailer? (Y/N) = > ");
textcolor(BLUE);
answer = toupper(getche());
getch();
if(answer == 'Y') {
gotoxy(22,14); cprintf("Enter the vehicle's weight (newtons) = > ");
wv =Get_input(100.0, 10000.0);
gotoxy(22,15); cprintf("Enter the trailer's weight (newtons) = > ");
wv =Get_input(100.0, 5000.0);
crscr();
Vs_max = Vs_max1(Vel, s1_max, dmax_l, g, wv, wt);
time1=0.0283*(Vs_max-Vel)/(dmax_l+g);
dee=(dmax_l+g)*9.81;
gotoxy(16,13); cprintf("The initial speed of the vehicle = > %2.1f km/hr", Vs_max);
gotoxy(16,14); cprintf("The total skidding time = > %2.2f seconds", time1);
gotoxy(16,15); cprintf("The rate of slowing or deceleration = > %2.1f meters/sec/sec", decrease);
}
else
if (answer == 'N') {
crscr();
wv=1.0;
wt=0.0;
Vs_max = Vs_max2(Vel, s1_max, dmax_l, g);
time1=0.0283*(Vs_max-Vel)/(dmax_l+g);
dee=(dmax_l+g)*9.81;
gotoxy(14,13); cprintf("The initial speed of the vehicle = > %2.1f km/hr", Vs_max);
gotoxy(14,14); cprintf("The total skidding time = > %2.2f seconds", time1);
gotoxy(14,15); cprintf("The rate of slowing or deceleration = > %2.1f meters/sec/sec", decrease);
}
} while ((answer != 'Y') && (answer != 'N'));
*p1 = Vs_max;
*p3 = g;
*s1 = s1_max;
*p5 = Vel1;
*p6 = time1;
curve=1;
*p10=curve;
void two_lane(float *p1, float *p2, float *p3, float *p4, float *p5, float *p6, float *p7, int *p10)
{
    float dece;
    int choice;
    char answer, user_choice[2];

    road_structure(); /* Displays a sub menu to show road elevation */
    clrscr();
    do
    {
        textcolor(YELLOW);
        gotoxy(15,10); cprintf("Vehicle state at the end of the skidmarks? Chose number below.");
        textcolor(BLUE);
        gotoxy(26,12); cprintf("1] Vehicle came to a complete stop. ");
        gotoxy(26,13); cprintf("2] Vehicle did not stop completely. ");
        textcolor(YELLOW);
        gotoxy(26,15); cprintf("Your choice is either (1 or 2) = > ");
        textcolor(BLUE);
        gets(user_choice);
        choice = atoi(user_choice);
        clrscr();
        if (choice == 1) { 
            Ve_max=0;
        }
        else
        if (choice == 2) { 
            textcolor(YELLOW);
            gotoxy(13,13); printf("Enter the final speed of the vehicle (km/hr) = > ");
            Ve_max = Get_input(1.0, 200.0);
        }
    } while ((choice != 1) && (choice != 2));
    clrscr();
    textcolor(YELLOW);
    textcolor(BLUE);
    gotoxy(16,12); cprintf("Were all wheels locked on both surfaces? (Y/N) = > ");
    fmax_1 = Get_input(0.01, 2.00);
    gotoxy(16,14); cprintf("Enter the friction coefficient of surface 2 = > ");
    fmax_2 = Get_input(0.01, 2.00);
    clrscr();

    do /* Loop to repeat response if a wrong entry is made */
    {
        textcolor(YELLOW);
        gotoxy(18,12); cprintf("Were all wheels locked on both surfaces? (Y/N) = > ");
        textcolor(BLUE);
        answer = toupper(getche());
        getch();
        if (answer == 'Y') {

    }
A-100

dmax_1 = fmax_1;
dmax_2 = fmax_2;
}
else
if (answer == 'N') {
    gotoxy(18,13); cprintf(" You need to calculate resultant drag factor"); */
    drag_factor2(fmax_1, fmax_2, g, Ve_max);
    return;
}
} while (answer != 'Y') && (answer != 'N');
clrscr();
textcolor(BLUE);
gotoxy(18,12); cprintf("Enter the skidmarks length on surface_1 (meters) = > ");
s1_max =Get_input(1.0, 2000.0);
gotoxy(18,14); cprintf("Enter the skidmarks length on surface_2 (meters) = > ");
s2_max =Get_input(0.0, 2000.0);
clrscr();
do /* Loop to repeat response if a wrong entry is made */
{
textcolor(YELLOW);
gotoxy(18,12); cprintf("Was the vehicle pulling a brakeless trailer? (Y/N) = > ");
textcolor(BLUE);
answer = toupper(getche());
if (answer == 'Y') {
gotoxy(22,14); cprintf("Enter the vehicle's weight (newtons) = > ");
wv =Get_input(100.0, 10000.0);
gotoxy(22,16); cprintf("Enter the trailer's weight (newtons) = > ");
w =Get_input(100.0, 5000);
gotoxy(14,15); cprintf("The initial speed of the vehicle = > %2.1f km/hr", Vs_max);
gotoxy(14,16); cprintf("The total skidding time = > %2.2f seconds", time);
gotoxy(14,17); cprintf("The rate of slowing or deceleration = > %2.1f meters/sec/sect", dece);
}
else
if (answer == 'N') {
    getchr;
wv = 1.0;
w = 0.0;
    Vs_max =Vs_max4(Ve_max,s1_max,s2_max,dmax_1,dmax_2,g);
    Ve1 =sqrt(Vs_max*Vs_max-253*s1_max*(dmax_1+g));
time1 = 0.0283*(Vs_max-Ve1)/(dmax_1+g);
time2 = 0.0283*(Ve1-Ve_max)/(dmax_2+g);
time = time1 + time2;
dece = 0.278*(Vs_max-Ve_max)/time;
gotoxy(14,15); cprintf("The initial speed of the vehicle = > %2.1f km/hr", Vs_max);
gotoxy(14,16); cprintf("The total skidding time = > %2.2f seconds", time);
gotoxy(14,17); cprintf("The rate of slowing or deceleration = > %2.1f meters/sec/sect", dece);
time = time1 + time2;
dece = 0.278 * (Vs_max - Ve_max) / time;
c1rschr();
gotoxy(14, 15); cprintf("The initial speed of the vehicle = > %2.1f km/hr", Vs_max);
gotoxy(14, 16); cprintf("The total skidding time = > %2.2f seconds", time);
gotoxy(14, 17); cprintf("The rate of slowing or deceleration = > %2.1f meters/sec/sec", dece);
}
} while ((answer != 'Y') && (answer != 'N'));
*f1 = dmax_1;
*f2 = dmax_2;
*p3 = g;
*p1 = Vs_max;
*s1 = s1_max;
*s2 = s2_max;
*p5 = Ve_max;
*p6 = time1;
*p7 = time2;
curve = 2;
*p10 = curve;

void three_lane(float *p1, float *f1, float *f2, float *f3, float *s1, float *s2, float *s3, float *p3, float *p5, float *p6, float *p7, float *p8, int *p10)
{
float dece, dmax;
int choice;
char answer, user_choice[2];

road_structure(); /* Displays s sub menu of road elevation */
c1rschr();
textcolor(YELLOW);
gotoxy(14, 11); cprintf("Skidding occurred on three different surfaces");
textcolor(BLUE);
gotoxy(14, 13); cprintf("Enter the friction coefficient of surface_1 = > ");
fmax_1 = Get_input(0.01, 2.00);
gotoxy(14, 15); cprintf("Enter the friction coefficient of surface_2 = > ");
fmax_2 = Get_input(0.01, 2.00);
gotoxy(14, 17); cprintf("Enter the friction coefficient of surface_3 = > ");
fmax_3 = Get_input(0.01, 2.00);
c1rschr();
textcolor(YELLOW);
do
{
textcolor(YELLOW);
gotoxy(15,12); cprintf("Vehicle state at the end of skidmarks. Chose number below");
gotoxy(26,14); printf(" 1] Vehicle came to a complete stop\n");
gotoxy(26,15); printf(" 2] Vehicle did not stop completely\n");
gotoxy(26,17); printf(" Your choice is (1 or 2) = > ");

gtextcolor(YELLOW);
gets(user_choice);
choice = atoi(user_choice);
clearscr();
if (choice = = 1) {
  Ve2 = 0;
} else
  if (choice = = 2) {
    gotoxy(22,14); printf("Enter the final speed of the vehicle (km/hr) = > ");
    Ve2 = Get_input(1.0, 200.0);
  }
} while ((choice != 1) && (choice != 2));
clearscr();
do /* Loop to repeat response if a wrong entry is made */
{
  textcolor(YELLOW);
  gotoxy(12,11); cprintf("Were all the wheels locked on all the three surfaces? (Y/N) = > ");
  textcolor(BLUE);
  answer = toupper(getche());
  getche();
  if (answer == 'Y') {
    dmax_1 = fmax_1;
    dmax_2 = fmax_2;
    dmax_3 = fmax_3;
  } else
    if (answer == 'N') {
      gotoxy(18,13); printf("You need to calculate the combined drag factor.\n");
      press_return();
      drag_factor2(fmax_1, fmax_2, g, Ve_max);
      return;
  }
} while ((answer != 'Y') && (answer != 'N'));
textcolor(BLUE);
gotoxy(12,13); cprintf("Enter the skidmarks length on surface_1 (meters) = > ");
s1_max = Get_input(1.0, 2000.0);
gotoxy(12,15); cprintf("Enter the skidmarks length on surface_2 (meters) = > ");
s2_max = Get_input(0.0, 2000.0);
gotoxy(12,17); cprintf("Enter the skidmarks length on surface_3 (meters) = > ");
s3_max = Get_input(0.0, 2000.0);
clearscr();
textcolor(YELLOW);
do  /* loop to repeat response if a wrong entry is made */
{
    textcolor(YELLOW);
    gotoxy(14,12); cprintf("Was the vehicle pulling a brakeless trailer? (Y/N) = > ");
    textcolor(BLUE);
    answer = toupper(getche());
    getch();
    if (answer == 'Y') {
        gotoxy(18,14); cprintf("Enter the vehicle's weight (newtons) = > ");
        wv = Get_input(100.0, 10000.0);
        gotoxy(18,15); cprintf("Enter the trailer's weight (newtons) = > ");
        wt = Get_input(100.0, 5000);
        clrscr();
        Vs_max = Vs_max7(Ve2,s1_max,s2_max,s3_max,dmax_1,dmax_2,dmax_3,g,wv,wt);
        Vel = sqrt(Vs_max*Vs_max-253*s1_max*(dmax_1+g));
        Ve_max = sqrt(Vel*Vel-253*s2_max*(dmax_2+g));
        time1 = 0.0283*(Vs_max-Vel)/(dmax_1+g);
        time2 = 0.0283*(Vel-Ve_max)/(dmax_2+g);
        time3 = 0.0283*(Ve_max-Ve2)/(dmax_3+g);
        time = time1 + time2 + time3;
        dece = 0.278*(Vs_max-Ve2)/time;
        gotoxy(16,13); cprintf("The initial speed of the vehicle = > %2.1f km/hr", Vs_max);
        gotoxy(16,14); cprintf("The total skidding time = > %2.2f seconds",time);
        gotoxy(16,15); cprintf("The rate of slowing or deceleration = > %2.1f meters/sec/sec. ",dece);
        getchar();
    } else
    if (answer == 'N') {
        clrscr();
        wv = 1.0;
        wt = 0.0;
        Vs_max = Vs_max8(Ve2,s1_max,s2_max,s3_max,dmax_1,dmax_2,dmax_3,g);
        Vel = sqrt(Vs_max*Vs_max-253*s1_max*(dmax_1+g));
        Ve_max = sqrt(Vel*Vel-253*s2_max*(dmax_2+g));
        time1 = 0.0283*(Vs_max-Vel)/(dmax_1+g);
        time2 = 0.0283*(Vel-Ve_max)/(dmax_2+g);
        time3 = 0.0283*(Ve_max-Ve2)/(dmax_3+g);
        time = time1 + time2 + time3;
        dece = 0.278*(Vs_max-Ve2)/time;
        gotoxy(16,13); cprintf("The initial speed of the vehicle = > %2.1f km/hr", Vs_max);
        gotoxy(16,14); cprintf("The total skidding time = > %2.2f seconds",time);
        gotoxy(16,15); cprintf("The rate of slowing or deceleration = > %2.1f meters/sec/sec. ",dece);
    }
} while ((answer != 'Y') && (answer != 'N'));
*p1 = Vs_max;
*f1 = dmax_1;
*f2 = dmax_2;
*f3 = dmax_3;
*s1 = s1_max;
*s2 = s2_max;
*s3 = s3_max;
*p3 = g;
*p5 = Ve_max;
*p6 = time1;
*p7 = time2;
*p8 = time3;
curve = 3;
*p10 = curve;

void drag_factor1(float *f1)
{
    float pr, fd_max, dmax;
    float ps, rd_max, x, y;
    int choice;
    char user_choice[2];

do
{
    textcolor(YELLOW);
    gotoxy(14,10); cprintf("Chose one the options below to find the total drag.");
    textcolor(BLUE);
    gotoxy(14,12); cprintf("1] To provide a reasonable estimate or use the rolling");
    gotoxy(14,13); cprintf(" friction (0.01) as the drag on the rolling wheels.");
    gotoxy(14,15); cprintf("2] To provide the percentage of the vehicle weight ");
    gotoxy(14,16); cprintf(" carried by the rolling wheels.");
    textcolor(YELLOW);
    gotoxy(26,18); cprintf("Your option is (1 or 2) = > ");
    gets(user_choice);
    choice = atoi(user_choice);
    clrscr();
    if (choice = 1) {
        gotoxy(14,14); cprintf("Enter the drag factor (or estimate) of the rolling wheels. = > ");
        rr = Get_input(0.01, 2.00);
        fd_max = rr;
        clrscr();
        gotoxy(18,14); cprintf("Enter the coefficient of friction of the road = > ");
        rd_max = Get_input(0.01, 2.00);
        clrscr();
        combine_drag(fd_max, rd_max, &dmax, &x, &y);
        gotoxy(15,12); cprintf("The longitudinal position of the center of mass = %2.2f meters", y);
        gotoxy(15,13); cprintf("Height of the center of mass of the vehicle = %2.2f meters", x);
        gotoxy(15,14); cprintf("The resultant drag factor = > %2.2f", dmax);
        dmax_1 = dmax;
    }
press_return();

} if (choice == 2) {
  gotoxy(10,14); cprintf("Enter the percentage of the vehicle weight on rolling wheel(s) => ");
  pr = Get_input(0.0, 100.0);
  ps = 100-pr;
  clrscr();
  gotoxy(18,12); printf("Enter the friction coefficient of the road => ");
  fmax_l = Get_input(0.01, 2.00);
  dmax_l = 2*(rr *(pr/100)) + 2*((ps/100) * fmax_l);
}
} while ((choice != 1) && (choice != 2));
*fl = dmax_1;

void drag_factor2(float fmax_1, float fmax_2, float g, float Ve_max)
{
  float rd_max;
  float fd_max;
  float dmax;
  float wv, wt,x,y;
  float dece;
  char answer, user_choice[2];
  int choice;

  do
  {
    textcolor(LIGHTCYAN);
    gotoxy(12,8); cprintf("Describe the path of the wheels during the sliding motion.");
    gotoxy(12,9); cprintf("Chose number below.");
    textcolor(BLUE);
    gotoxy(12,11); cprintf("1\] One front wheel slid on pavement_1 and three on pave_2.");
    gotoxy(12,12); cprintf("2\] One rear wheel slid on pavement_1 and three on pave_2.");
    gotoxy(12,13); cprintf("3\] Two front wheels slid on pave_1 and two rear on pave_2");
    gotoxy(12,14); cprintf("4\] Right front and right rear on pave_1 and left front and");
    gotoxy(12,15); cprintf(" left rear on pave_2.");
    textcolor(YELLOW);
    gotoxy(18,17); cprintf("Your choice is (1, 2, 3 or 4) => ");
    textcolor(BLUE);
    gets(user_choice);
    choice = atoi(user_choice);
    clrscr();
    textcolor(YELLOW);
    if (choice == 1) {
      rd_max = fmax_2;
    }
  } while ((choice != 1) && (choice != 2));
fd_max = (fmax_1 + fmax_2)/2;
if (rd_max != fd_max) {
    gotoxy(16,12); cprintf("Rear axle drag is not equal to the front axle drag");
    combine_drag(fd_max, rd_max, &dmax,&x,&y);
    textcolor(LIGHTCYAN);
}
else {
    gotoxy(18,12); printf("Rear wheels drag factor equals front wheels drag\n");
    dmax = (fd_max + rd_max)/2;
}

} else
if (choice==2) {
    gotoxy(16,11); cprintf("Front wheels drag is equal to the fric. coeff of pave_2");
    fd_max = fmax_2;
    rd_max = (fmax_1 + fmax_2)/2;
    if (rd_max != fd_max) {
        gotoxy(16,12); cprintf("Rear axle drag is not equal to the front axle drag");
        combine_drag(fd_max, rd_max, &dmax,&x,&y);
    }
    else {
        gotoxy(16,12); cprintf("Rear axle drag factor equals front axle drag");
        dmax = (fd_max + rd_max)/2;
    }
}
else
if (choice==3) {
    gotoxy(18,11); printf("Front axle drag is equal to the friction coeff. of pave_1");
    gotoxy(18,12); printf("Rear axle drag is equal to the friction coeff. of pave_2");
    fd_max = fmax_1;
    rd_max = fmax_2;
    if (rd_max != fd_max) {
        gotoxy(18,13); printf("Rear axle drag is not equal to the front axle drag");
        combine_drag(fd_max, rd_max, &dmax,&x,&y);
    }
    else {
        gotoxy(18,12); printf("Rear axle drag factor equals to front axle drag");
        dmax = (fd_max + rd_max)/2;
        press_return();
    }
}
else
if (choice==4) {
    gotoxy(20,14); cprintf("Front axle drag equals rear axle drag");
    fd_max = (fmax_1 + fmax_2)/2;
    rd_max = fd_max;
    dmax = (fmax_1 + fmax_2)/2;
}
} while ((choice != 1) && (choice != 2) && (choice != 3) && (choice != 4));
clrscr();
gotoxy(16,12); cprintf("Enter the length of the skidmarks (meters) = > ");
s1_max = Get_input(1.0, 2000.0);
crscr();
do
{
   textcolor(YELLOW);
gotoxy(18,12); cprintf("Was the vehicle pulling a brakeless trailer? (Y/N) = > ");
textcolor(BLUE);
anwer = toupper(getche());
if (answer == 'Y') {
gotoxy(22,14); printf("Enter the vehicle's weight (newtons) = > ");
wv = Get_input(100.0, 10000.0);
gotoxy(22,16); printf("Enter the trailer's weight (newtons) = > ");
w = Get_input(100.0, 5000);
Vs_max = Vs_max11(Ve_max, s1_max, dmax, g, wv, w);
}
else
   if (answer == 'N') {
   wv=1.0;
w=0.0;
   Vs_max = Vs_max12(Ve_max, s1_max, dmax, g);
   }
} while (answer != 'Y') && (answer != 'N');
crscr();
textcolor(LIGHTCYAN);
time = (7.20*s1_max)/(Vs_max + Ve_max);
dece = 0.278*(Vs_max - Ve_max)/time;
gotoxy(15,12); cprintf("The rear axle drag factor = > %f", rd_max);
gotoxy(15,13); printf("The front axle drag factor = > %f", fd_max);
gotoxy(15,14); printf("The resultant drag factor = > %f", dmax);
gotoxy(15,16); cprintf("The initial speed of the vehicle = > %2.1f km/h", Vs_max);
gotoxy(15,17); cprintf("The total skidding time = > %2.2f seconds", time);
gotoxy(15,18); cprintf("The rate of slowing or deceleration = > %2.1f meters/sec/sec.", dece);
getche();
}

void combine_drag(float fd_max, float rd_max, float *p5, float *p8, float *p9)
{
   /* Function determines the resultant drag factor */
   float s1_max;
   float L;
   float R;
   float W;
   float Wr;
   float Wh;
   float h;
float x;
float y;
float dmax;
char answer;

textcolor(LIGHTCYAN);
gotoxy(14, 8); cprintf("To find the combine or resultant drag factor, ");
gotoxy(14, 9); cprintf("you need to have values of the following parameters.");
do
{
    textcolor(BLUE);
gotoxy(14, 11); cprintf("Height of center of mass of the vehicle");
gotoxy(14, 12); cprintf("Length of the vehicle's wheelbase");
gotoxy(14, 13); cprintf("Weight on the rear axle with vehicle level");
gotoxy(14, 14); cprintf("Weight of the rear axle when the front is hoisted h ft");
gotoxy(14, 15); cprintf("Weight of the vehicle");
gotoxy(14, 16); cprintf("Radius of wheel with tire");
/* loop to repeat response if a wrong entry is made */
textcolor(YELLOW);
gotoxy(14, 18); cprintf("Can you get the values for all the above information? (Y/N) = > ");
textcolor(BLUE);
an = toupper(getche());
clrscr();
textcolor(YELLOW);
if (answer = = 'Y') {
    gotoxy(13, 7); cprintf("What is the length of vehicle's wheelbase (meters) = > ");
    L = Get_input(1.0, 50.0);
    gotoxy(13, 9); cprintf("What is the radius of wheel with tires (meters) = > ");
    R = Get_input(0.5, 10.0);
    gotoxy(13, 11); cprintf("What is the weight of vehicle (newtons) = > ");
    W = Get_input(100.0, 10000.0);
    gotoxy(13, 13); cprintf("Weight on the rear axle with vehicle level (newtons) = > ");
    Wr = Get_input(50.0, 5000);
    gotoxy(13, 15); cprintf("Weight of rear axle when the front end is hoisted 'h meters' = > ");
    Wh = Get_input(50.0, 5000);
    gotoxy(13, 17); cprintf("Height to which an axle is hoisted (meters) = > ");
    h = Get_input(0.5, 10.0);
    y = R / L + ((Wh - Wr) * (L - h) / (h * W));
    dmax = (fd_max * (1 - y) + rd_max * y) / (1 - y * (fd_max - rd_max));
    *p5 = dmax;
    *p8 = x;
    *p9 = y;
    clrscr();
}
else
if (answer = = 'N') {
    textcolor(LIGHTCYAN);
}
gotoxy(18,12); cprintf("The information is needed to find the combine drag factor.");
gotoxy(18,13); cprintf("before you can find the initial speed of vehicle");
gotoxy(18,14); cprintf("Press ENTER/RETURN key to return to Speed Menu.");
getchar();
speed();
}
} while ((answer! = 'Y') && (answer! = 'N'));

void plotgraph(float xmin,float xmaximum[],float xma,float yma,float yy,int num,int curve)
{
    float result,x,y,gir,result1;
    float xs,ys,xsp,ysp;
    int x_real,y_real,i,case, icounter, iabbas;
    int i,maxx,maxy,j,k, counter,counter1;
    FILE *ioptr1;
    FILE *ioptr2;
    FILE *ioptr3;
    int GraphDriver, GraphMode;
    float nur1,nur2,nur3,eric,nur;
    static int hadi[] = {0, 1, 16, 17};
    static int deen[] = {0, 2, 24, 25};
    static int nuruj[] = {0, 3, 26, 27};

    if ((num == 1) || (num== 16) || (num== 17)) {
        ioptr1 = fopen("mydat1.dat", "w");
        ioptr2 = fopen("mydat2.dat", "w");
        ioptr3 = fopen("mydat3.dat", "w");
        maxx = getmaxx();
        maxy = getmaxy();
        x=xmin;
        y=yy;
        for (j=1; j<=curve; j++) {
            do
            {
                ys=(maxy-60)/yma;
                xs=(maxx-60)/xma;
                if (j==1) {
                    result = my_function(hadi[j],x,y,dmax_1,dmax_2, dmax_3);
                    gir = result;
                    fprintf(ioptr1,"\n f %f *",y,gir);
                }
                else if (j==2) {
                    result = my_function(hadi[j],x,y,dmax_1,dmax_2, dmax_3);
                    fprintf(ioptr2,"\n f %f *",&,x,result);
                }
            }
        }
    }
}
} 
else if (j == 3) {
    result = my_function(hadi[j], x, y, dmax_1, dmax_2, dmax_3);
    result1 = result;
    fprintf(ioptr3, "\n %f %f", x, result1);
}

xsp = x * xs;
ysp = result * ys;
_ _real = 60 + xsp;
y_real = 419 - ysp;
putpixel(x_real, y_real, IS);
x = x + 0.01;
} while (x < xminaxum[j]);
}

getch();
closegraph();
restorecrtmode();
close(ioptr1);
close(ioptr2);
close(ioptr3);
detectgraph(&GraphDriver, &GraphMode);
initgraph(&GraphDriver, &GraphMode, "");
ioptr1 = fopen("mydat1.dat", "r");
ioptr2 = fopen("mydat2.dat", "r");
ioptr3 = fopen("mydat3.dat", "r");
textmode(3);
setbkcolor(7);
textcolor(1);
textbackground(7);
if (curve == 1) {
    gotoxy(30, 2); cprintf("friction coeff = %2.2f", dmax_1);
    gotoxy(30, 3); cprintf("percent grade = %1.2f", g);
    gotoxy(30, 5); cprintf("BRAKING SURFACE 1");
    gotoxy(30, 6); cprintf("DISTANCE SPEED");
    gotoxy(30, 7); cprintf("(meters) (km/h)");
    gotoxy(30, 8); cprintf("-------------------
");
    counter1 = 0;
    icounter = 1;
    while (!feof(ioptr1)) {
        for (i = 1; j <= icounter; j = j + 1)
            fscanf(ioptr1, "%f %f", &eric, &nurl);
            cprintf("%3.0f %7.1f%n", eric, nurl);
    counter = counter1 + 5;
    icounter = counter * 10;
    scroll();
    }
}
else if (curve == 2) {
    gotoxy(24, 2); cprintf("friction coeff. of surface 1 = %2.2f", dmax_1);
    gotoxy(24, 3); cprintf("friction coeff. of surface 2 = %2.2f", dmax_2);
    gotoxy(24, 4); cprintf("percent grade of surfaces = %2.2f", g);
}
gotoxy(20,7); cprintf("BRAKING SURFACE 1 SURFACE 2");
gotoxy(20,8); cprintf("DISTANCE SPEED SPEED");
gotoxy(20,9); cprintf("(meters) (km/hr) (km/hr)");
gotoxy(20,10); cprintf("-----------------------------\n");
counter1=0;
icounter=1;
while(!feof(ioptr1)) {
    for (j = 1; j <= icounter; j = j + 1)
        fscanf(ioptr1,"%f %f", &eric,&nurl);
    counter =counter1 + 5;
icounter=counter*10;
cprintf(" %23.0f %7.1f\n", eric,nurl);
    scroll();
}
while(!feof(ioptr2)) {
    for (j = 1; j <= icounter; j = j + 1)
        fscanf(ioptr2,"%f %f", &eric,&nur2);
cprintf(" %23.0f %7.1f\n", eric,nur2);
    counter =counter1 + 5;
icounter=counter*10;
    scroll();
}
}
else if (curve == 3) {
    gotoxy(20,3); cprintf("friction coeff. of surface 1 = %2.2f", dmax_1);
gotoxy(20,4); cprintf("friction coeff. of surface 2 = %2.2f", dmax_2);
gotoxy(20,5); cprintf("friction coeff. of surface 3 = %2.2f", dmax_3);
gotoxy(20,6); cprintf("percent grade of surfaces = %2.2f\%\n", g);
gotoxy(10,9); cprintf("BRAKING SURFACE 1 SURFACE 2 SURFACE 3");
gotoxy(20,10); cprintf("DISTANCE SPEED SPEED SPEED");
gotoxy(20,11); cprintf("(meters) (km/hr) (km/hr) (mph)\n");
gotoxy(20,12); cprintf("-----------------------------\n");
counter1=0;
icounter=1;
while(!feof(ioptr1)) {
    for (j = 1; j <= icounter; j = j + 1)
        fscanf(ioptr1,"%f %f", &eric,&nurl);
cprintf(" %13.0f %7.1f\n", eric,nurl);
    counter =counter1 + 5;
icounter=10*counter;
    scroll();
}
while(!feof(ioptr2)) {
    for (j = 1; j <= icounter; j = j + 1)
        fscanf(ioptr2,"%f %f", &eric,&nur2);
cprintf(" %13.0f %7.1f\n", eric,nur2);
    counter =counter1 + 5;
icounter=10*counter;
    scroll();
}
while(!feof(ioptr3)) {
    for (j = 1; j <= icounter; j++) {
        fscanf(ioptr3, "%f %f", &eric, &nur3);
        cprintf("%13.0f", eric, nur3);
        counter = counter + 1;
        icounter = 10 * counter;
    }
}
press_return();
closegraph();
restorecrtmode();
fclose(ioptr1);
fclose(ioptr2);
fclose(ioptr3);
}
else if ((num == 2) || (num == 24) || (num == 25) || (num == 19)) {
    ioptr1 = fopen("mydat1.dat", "w");
    maxx = getmaxx();
    maxy = getmaxy();
    x = xmin;
    y = yy;
    for (j = 1; j <= curve; j++) {
        do {
            ys = (maxy - 60) / yma;
            xs = (maxx - 60) / xma;
            result = my_function(deen[j], x, y, dmax_1, dmax_2, dmax_3);  
            fprintf(ioptr1, "\n %f %f", x, result);
            xsp = x * xs;
            ysp = result * ys;
            x_real = 60 + xsp;
            y_real = 419 - ysp;
            putpixel(x_real, y_real, 15);
            x = x + 0.01;
            y = y + 0.01;
        } while (x < xmaximum[j]);
    }
getch();
closegraph();
restorecrtmode();
fclose(ioptr1);
detectgraph(&GraphDriver, &GraphMode);
initgraph(&GraphDriver, &GraphMode, "");
ioptr1 = fopen("mydat1.dat", "r");
textmode(3);
setbkcolor(7);
textcolor(1);
textbackground(7);
gotoxy(22, 3); cprintf("friction coeff. of surface 1 = %2.2f", dmax_1);
gotoxy(22,4); cprintf("friction coeff. of surface 2 = %2.2f", dmax_2);
gotoxy(22,5); cprintf("friction coeff. of surface 3 = %2.2f", dmax_3);
gotoxy(22,6); cprintf("percent grade of surfaces = %2.2f", g);
gotoxy(30,8); cprintf("BRAKING BRAKING");
gotoxy(30,9); cprintf(" TIME SPEED");
gotoxy(30,10); cprintf("(secs) (km/hr)";
gotoxy(30,11); cprintf("------------------\n");
counter = iabbas=0;
counter1 =0;
icounter =1;
while(!feof(ioptr1))
{
    for (j=1; j <= icounter; j++)
        fscanf(ioptr1, "%f %f", &eric,&nur1);
        cprintf("%33.2f %7.1f\n", eric,nur1);
    counter =counter1 +5;
icounter=counter*10;
    scroll();
}
pres_return();
closegraph();
restorecrtmode();
fclose(ioptr1);
}
else
if ((num = =3) || (num = =26) || (num = =27))
{
    ioptr1 = fopen("mydat1.dat", "w");
    maxx = getmaxx();
    maxy = getmaxy();
x=xmin;
y=yy;
for (j =1; j = curve; j++)
{
do
{
    ys=(maxy-60)/yma;
x=(maxx-60)/xma;
    result= my_function(nuru[j],x,y,dmax_1,dmax_2, dmax_3);
    fprintf(ioptr1,"\n %f %f",x,result);
    xsp=x*xs;
ysp=result*ys;
x_real=60+xsp;
y_real=419-ysp;
putpixel(x_real,y_real,15);
x=x+0.01;
} while (x < xmaximum[j]);
}
getch();
closegraph();
restorecrtmode();
fclose(ioptr1);
detectgraph(&GraphDriver, &GraphMode);
initgraph(&GraphDriver, &GraphMode, "");
ioptrl = fopen("mydat1.dat", "r");
textmode(3);
setbkcolor(7);
textcolor(1);
textbackground(7);
gotoxy(22,3); cprintf("friction coeff. of surface 1 = %2.2f", dmax_1);
gotoxy(22,4); cprintf("friction coeff. of surface 2 = %2.2f", dmax_2);
gotoxy(22,5); cprintf("friction coeff. of surface 3 = %2.2f", dmax_3);
gotoxy(22,6); cprintf("percent grade of surfaces = %2.2f", g);
gotoxy(30,9); cprintf("BRAKING BRAKING");
gotoxy(30,10); cprintf(" TIME DISTANCE");
gotoxy(30,11); cprintf("(secs) (meters)");
gotoxy(30,12); cprintf("----------------------
");
counterl =0;
counter = iabbas =0;
icounter = 1;
while(!feof(ioptrl)) {
   for (j = 1; j <= icounter; j++)
      fscanf(ioptrl, "%f %f", &eric, &nurl);
      cprintf(" %33.2f %7.0f\n", eric, nurl);
      counter=counterl +5;
icounter = 10*counter;
scroll();
}
press_return();
closegraph();
restorecrtmode();
close(ioptrl);

else if (num == 9) {
ioptrl = fopen("mydat1.dat", "w");
maxx = getmaxx();
maxy = getmaxy();
x = xmin;
y = yy;
for (j=1; j <= curve; j++) {
do
   
   ys=(maxy-60)/yma;
   xs=(maxx-60)/xma;
   result = my_function(num,x,y,dmax_1,dmax_2, dmax_3);
   fprintf(ioptrl, "%n %f %f", x, result);
   xsp=x*xs;
   ysp=result*ys;
   x_real=60+xsp;
   y_real=419-ysp;
   putpixel(x_real,y_real,15);
\[ x = x + 0.01; \]

\} while (x < xmaximum[j]);

}  

getch();

closegraph();

restorecrtmode();

fclose(ioptr1);

detectgraph(&GraphDriver, &GraphMode);

initgraph(&GraphDriver, &GraphMode, "");

ioptr1 = fopen("mydat1.dat", "r");

textmode(3);

setbkcolor(7);

textcolor(1);

textbackground(7);

gotoxy(30,3); cprintf("friction coefficient = %2.2f", dmax_l);

gotoxy(30,4); cprintf("percent grade of road = %2.2f", g);

gotoxy(30,6); cprintf("BRAKING BRAKING");

gotoxy(30,7); cprintf(" TIME DISTANCE");

gotoxy(30,8); cprintf("(secs) (meters)\n");

gotoxy(30,9); cprintf("---------------------- \n");

counter = iabbas = 0;

icounter = 0;

icounter = 1;

while (!feof(ioptr1)) {

for (j = 1; j <= icounter; j++)

fscanf(ioptr1, "%f %f", &eric, &nurl);

cprintf(" %33.2f % 7 .2f\n", eric, nurl);

counter = counter + 5;

icounter = 10 * counter;

scroll();

}  

press_return();

closegraph();

restorecrtmode();

close(ioptr1);

} else {

ioptr1 = fopen("mydat1.dat","w");

maxx = getmaxx();

maxy = getmaxy();

for (j = 1; j <= curve; j++) {

x = xmin;

y = yy;

do

{  

ys = (maxy - 60) / yma;

xs = (maxx - 60) / xma;

result = my_function(num, x, y, dmax_1, dmax_2, dmax_3);

fprintf(ioptr1, "\n%f %f", x, result);

xsp = x * xs;

} while (x < xmaximum[j]);
ysp = result*ys;  
x _real=60+xsp;  
y _real=419-ysp;  
putpixel(x_real,y_real, 15);  
x=x+0.1;  
} while (x < xmaximum[j]);  
}  
getch();  
closegraph();  
restorecrtcmode();  
fclose(ioptrl);  
detectgraph(&GraphDriver, &GraphMode);  
initgraph(&GraphDriver, &GraphMode,"" );  
ioptrl = fopen("mydat1.dat", "r" );  
textmode(3);  
setbkcolor(7);  
textcolor(1);  
textbackground(7);  
if ( ((num = =4) | |(num = =8)) ) {  
gotoxy(30,3); cprintf("friction coefficient = %.2f",dmax_1);  
gotoxy(30,4); cprintf("percent grade of road = %.2f", g);  
gotoxy(30,7); cprintf(" END BRAKING");  
gotoxy(30,8); cprintf("SPEED DISTANCE");  
gotoxy(30,9); cprintf("(km/h) (meters)" );  
gotoxy(30,10); cprintf("------------------------");  
}  
else if (num = =12) {  
gotoxy(30,2); cprintf("friction coefficient = %.2f",dmax_1);  
gotoxy(30,3); cprintf("percent grade of road = %.2f", g);  
gotoxy(30,4); cprintf("initial speed = %.2f mph", Vs_max);  
gotoxy(30,6); cprintf("BRAKING BRAKING");  
gotoxy(30,7); cprintf(" SPEED TIME");  
gotoxy(30,8); cprintf("(km/hr) (secs)" );  
gotoxy(30,9); cprintf("------------------------");  
}  
else if (num = =5) {  
gotoxy(30,2); cprintf("friction coefficient = %.2f",dmax_1);  
gotoxy(30,3); cprintf("percent grade of road = %.2f", g);  
gotoxy(30,4); cprintf("initial speed = %.2f mph", Vs_max);  
gotoxy(30,6); cprintf("BRAKING BRAKING");  
gotoxy(30,7); cprintf("DISTANCE TIME");  
gotoxy(30,8); cprintf("(meters) (secs)" );  
gotoxy(30,9); cprintf("------------------------");  
}  
counter = iabbras =0;  
counter1 =0;  
icounter= 1;  
while (!feof(ioptrl)) {  
for (j =1; j <= icounter; j + +)  
 fscanf(ioptrl,"%f %f", &eric,&nur1);
cprintf("%33.2f %7.2f\n", eric, nur1);
counter = counter1 + 5;
icounter = 10 * counter;
scroll();
}
press_return();
closegraph();
restorecrtmode();
fclose(ioptr1);
}

float my_function(int num, float x, float y, float dmax_1, float dmax_2, float dmax_3)
{
    if (num == 1) {
        return(sqrt(Vs_max * Vs_max - (253 * x * (wv / (wv + wt)) * (dmax_1 + g))));
    }
    else if (num == 2) {
        return(Vs_max - (35.3 * x * (wv / (wv + wt)) * (dmax_1 + g)));
    }
    else if (num == 3) {
        return(0.278 * x * Ve1 + 16.1 * x * x * (wv / (wv + wt)) * (dmax_1 + g));
    }
    else if (num == 4) {
        return(0.00393 * ((Vs_max * Vs_max) - (x * x)) / (dmax_1 + g));
    }
    else if (num == 5) {
        return(sqrt(0.204 * x / (dmax_1 + g) + 0.0008 * Ve_max * Ve_max / ((dmax_1 + g) * (dmax_1 + g))) - 0.0283 * Ve_max / (dmax_1 + g));
    }
    else if (num == 6) {
        return(0.0283 * (Vs_max - x) / (dmax_1 + g));
    }
    else if (num == 7) {
        return(0.452 * (sqrt(s1_max / (x + g))));
    }
    else if (num == 8) {
        return(0.00393 * ((Vs_max * Vs_max) - (x * x)) / (wv / (wv + wt)) * (dmax_1 + g));
    }
}
else
  if (num == 9) {
    return(0.278*x*Ve_max + 16.1*x*x*(wv/(wv+wt))*(dmax_1 + g));
  }
else
  if (num == 12) {
    return((0.0283*(Vs_max-x))/((wv/(wv+wt))*(dmax_1 + g)));
  }
else
  if (num == 13) {
    return(sqrt((0.0204*x /(dmax_1 + g) + 0.0008* Ve_max* Ve_max/ ((dmax_1 + g)* (dmax_1 + g)))- 0.0283* Ve_max/(dmax_1 + g));
  }
else, if (num == 15) {
    return(sqrt(Vs_max*Vs_max- (253*x*(wv/(wv+wt))*(dmax_1 + g))));
  }
else
  if (num == 16) {
    return(sqrt(Vs_max* Vs_max- (253*(s1_max* (wv/(wv+wt)*) (dmax_1 + g)- s1_max* (wv/(wv+wt))* (dmax_2+g) + x*(wv/(wv+wt))*(dmax_2+g)))));
  }
else
  if (num == 17) {
    return(sqrt(Vs_max* Vs_max- (253* (s1_max* (wv/(wv+wt)) * (dmax_1 + g)- s1_max* (wv/(wv+wt))*(dmax_2+g) + (s1_max + s2_max)* (wv/(wv+wt))*(dmax_2-dmax_3)+ x*(wv/(wv+wt))*(dmax_3+g)))));
  }
else
  if (num == 18) {
    return(sqrt(y*y-(253*x*(dmax_1 + g))));
  }
else
  if (num == 19) {
    return(y-(35.3*x*(dmax_1 + g)));
  }
else
  if (num == 24) {
    return(Vs_max- (35.3* (time* (wv/(wv+wt))*(dmax_1 + g)- time* (wv/(wv+wt))*(dmax_2+g) + x*(wv/(wv+wt))*(dmax_2+g))));
  }
else
  if (num == 25) {
    return(Vs_max- (35.3* (time* (wv/(wv+wt))*(dmax_1 + g)- time* (wv/(wv+wt))*(dmax_2+g) + (time + time2)* (wv/(wv+wt))*(dmax_2-dmax_3)+ x*(wv/(wv+wt))*(dmax_3+g))));
  }
else
  if (num == 26) {
    return(0.278*time*Ve_l + 4.91*time*time*((wv/(wv+wt))*(dmax_1 + g) + 0.278*(x-time)*
Ve_max + 4.91*(x-time1)*(x-time1)*(wv/(wv+wt))*(dmax_2+g);

else
if (num == 27) {
    return(0.278* time1* Ve1 + 4.91* time1* time1* (wv/(wv+wt))*( dmax_1 + g) + 0.278*time2*
    Ve_max + 4.91*time2*time2*(wv/(wv+wt))*(dmax_2 + g) + 0.278*(x-time1-time2)*Ve2 + 4.91* 
    (x-time1-time2)* (x-time1-time2)* (wv/(wv+wt))*(dmax_3 + g));
}

float yscale(num)
{
    ((num == 5) || (num == 6) || (num == 7) || (num == 10) || (num == 13) || (num == 14))
    {
        return(10);
    }
    else
if ((num == 3) || (num == 4) || (num == 8) || (num == 9) || (num == 26) || (num == 27))
    {
        return(400);
    }
else
if ((num == 1) || (num == 2) || (num == 15) || (num == 16) || (num == 17) || (num == 18) 
    || (num == 19) || (num == 24) || (num == 25))
    {
        return(200);
    }

float xscale(int num)
{
    if ((num == 1) || (num == 5) || (num == 13) || (num == 14) || (num == 15) || (num == 18) ||
    (num == 22) || (num == 16) || (num == 17))
    {
        return(400);
    }
else
if ((num == 2) || (num == 3) || (num == 9) || (num == 19) || (num == 24) || (num == 25) ||
    (num == 26) || (num == 27))
    {
        return(10);
    }
else
if ((num == 4) || (num == 6) || (num == 8) || (num == 11) || (num == 12) || (num == 15) || (num == 10))
    {
        return(200);
    }
else
    if (num == 7) return(1.0);
void calculator()
{
    float vs1, vs2, vs3, vs;
    float f1, f2, f3;
    float s1, s2, s3, s;
    float t1, t2, t3, t;
    float speed;
    float friction, f;
    float distance;
    float time;
    int choice, repeat;
    char user_choice[2], answer;

    textcolor(YELLOW);
    gotoxy(24, 11); cprintf("Given any two of the input parameters");
    textcolor(BLUE);
    gotoxy(38, 12); cprintf("* speed");
    gotoxy(38, 13); cprintf("* distance");
    gotoxy(38, 14); cprintf("* friction coefficient");
    gotoxy(38, 15); cprintf("* time");
    textcolor(YELLOW);
    gotoxy(14, 16); cprintf(" the Calculator can compute the remaining two parameters.");
    press_return();
    textcolor(YELLOW);
    repeat = TRUE;
    while (repeat)
    {
        do
        {
            clrscr();
            gotoxy(18, 9); cprintf("Which pair of parameters are known? Chose number below");
            textcolor(BLUE);
            gotoxy(30, 11); cprintf(" 1] speed and distance");
            gotoxy(30, 12); cprintf(" 2] speed and friction coefficient");
            gotoxy(30, 13); cprintf(" 3] speed and time");
            gotoxy(30, 14); cprintf(" 4] distance and friction coefficient");
            gotoxy(30, 15); cprintf(" 5] distance and time");
            gotoxy(30, 16); cprintf(" 6] friction coefficient and time");
        }
    }

    do
    {
        textcolor(YELLOW);
        gotoxy(22, 18); cprintf("Your choice is (1, 2, 3, 4, 5 or 6) = > ");
        textcolor(BLUE);
        gets(user_choice);
        choice = atoi(user_choice);
        if (choice == 1) {
            clrscr();
            textcolor(YELLOW);
gotoxy(24,12); cprintf("Enter the vehicle’s speed (km/hr) = > ");
textcolor(BLUE);
vs = Get_input(5.0, 200.0);
textcolor(YELLOW);
gotoxy(24,14); cprintf("Enter the skidding distance (meters) = > ");
textcolor(BLUE);
s = Get_input(1.0, 500.0);
friction = f1(vs, s);
time = t1(vs, s);
gotoxy(24,16); cprintf("Friction coefficient = %2.2f", friction);
gotoxy(24,17); cprintf("Skidding Time = %2.2f seconds", time);
getchar();
press return();
}

else
if(choice == 2) {
clrscr();
gotoxy(24,12); cprintf("Enter the vehicle’s speed (km/hr) = > ");
vs = Get_input(5.0, 200.0);
gotoxy(24,14); cprintf("Enter the friction coefficient = > ");
f = Get_input(0.01, 2.00);
distance = s1(vs, f);
time = t2(vs, f);
gotoxy(24,16); cprintf("Skidding distance = %2.2f meters", distance);
gotoxy(24,17); cprintf("Skidding Time = %2.2f seconds", time);
getchar();
press return();
}
else
if(choice == 3) {
clrscr();
gotoxy(24,12); cprintf("Enter the vehicle’s speed (km/hr) = > ");
vs = Get_input(5.0, 200.0);
gotoxy(24,14); cprintf("Enter the skidding time (secs) = > ");
t = Get_input(0.5, 60.0);
distance = s2(vs, t);
friction = f2(vs, t);
gotoxy(24,16); cprintf("Skidding distance = %2.2f meters", distance);
gotoxy(24,17); cprintf("friction coefficient = %2.2f", friction);
getchar();
press return();
}
else
if(choice == 4) {
clrscr();
gotoxy(24,12); cprintf("Enter skidding distance (meters) = > ");
s = Get_input(5.0, 500.0);
gotoxy(24,14); cprintf("Enter the friction coefficient = > ");
f = Get_input(0.01, 1.25);
speed = vs1(s, f);
time = t3(s, f);
gotoxy(24,16); cprintf("Vehicle speed = %2.2f km/hr", speed);
gotoxy(24,17); cprintf("Skidding time = %2.2f seconds", time);
getchar();
press_return();
}
else
if (choice == 5) {
    clrscr();
gotoxy(24,12); cprintf("Enter the skidding distance (meters) = > ");
s = Get_input(5.0, 500.0);
gotoxy(24,14); cprintf("Enter the skidding time (sees) = > ");
t = Get_input(0.5, 60.0);
speed = vs2(s, t);
friction = D(s, t);
gotoxy(24,16); cprintf("Vehicle speed = %2.2f km/hr", speed);
gotoxy(24,17); cprintf("friction coefficient = %2.2f", friction);
getchar();
press_return();
}
else
if (choice == 6) {
    clrscr();
gotoxy(24,12); cprintf("Enter the friction coefficient = > ");
f = Get_input(0.01, 1.25);
gotoxy(24,14); cprintf("Enter the skidding time (sees) = > ");
t = Get_input(0.5, 60.0);
speed = vs3(f, t);
distance = s3(f, t);
gotoxy(24,16); cprintf("Vehicle speed = %2.2f km/hr", speed);
gotoxy(24,17); cprintf("Skid distance = %2.2f meters", distance);
getchar();
press_return();
}
} while ((choice != 1) && (choice != 2) && (choice != 3) && (choice != 4) && (choice != 5) && (choice != 6));
gotoxy(16,13); cprintf("Do you want to continue with the calculator? (Y/N) = > ");
answer = toupper(getch());
clrscr();
if (answer == 'Y') {
    repeat = TRUE;
}
else
if (answer == 'N') {
    repeat = FALSE;
    clrscr();
gotoxy(30,13); cprintf("This ends the session on Calculator");
textcolor(RED);
gotoxy(24,14); cprintf("Hit the ENTER/RETURN key to return to Main Menu");
getchar();
}
void press_return(void)
{
    gotoxy(27,25);
    textcolor(YELLOW);
cprintf("press\_RETURN- to continue");
gchar();
textcolor(WHITE);
c1rscr();
}

void downhill()
{
float gr;

textcolor(1);
gotoxy(20,12); cprintf("The grade of the road is negative. ");
gotoxy(20,13); printf("Enter the grade of the road (%) = ");
gr\= Get\_input(-30, 30);
g = gr/100;
c1rscr();
}

void uphill()
{
float gr;

textcolor(1);
gotoxy(20,12); cprintf("The grade of the road is positive. ");
gotoxy(20,13); printf("Enter the grade of the road (%) = ");
gr\= Get\_input(-30, 30);
g = gr/100;
c1rscr();
}
void auto_initialization(void)
{
    int graph_driver;
    int graph_mode;
    graph_driver = DETECT;
    initgraph(&graph_driver, &graph_mode, "");
}

void scroll()
{
    if (iabbas == 14) {
        iabbas = 0;
        press_return();
        printf("\n\n\n");
        setbkcolor(7);
        textcolor(1);
        textbackground(7);
    }
    else
        iabbas+ +;
}
#define Vs_max1(Ve_max, s1_max, dmax_1, g, wv, wt) \( \sqrt{254.45 \cdot s1_{\text{max}} \cdot \left(\frac{wv}{wv + wt}\right) \cdot \text{dmax}_1 + g} + \text{Ve_max} \cdot \text{Ve_max} \)

#define Vs_max2(Ve_max, s1_max, dmax_1, g) \( \sqrt{254.45 \cdot s1_{\text{max}} \cdot \text{dmax}_1 + g} + \text{Ve_max} \cdot \text{Ve_max} \)

#define Vs_max3(Ve_max, s1_max, s2_max, dmax_1, dmax_2, g, wv, wt) \( \sqrt{254.45 \cdot \left(\frac{wv}{wv + wt}\right) \cdot \left(\text{sl}_{\text{max}} \cdot \text{dmax}_1 + \text{s2}_{\text{max}} \cdot \text{dmax}_2\right) + g \cdot \left(\text{s1}_{\text{max}} + \text{s2}_{\text{max}}\right)} + \text{Ve_max} \cdot \text{Ve_max} \)

#define Vs_max4(Ve_max, s1_max, s2_max, dmax_1, dmax_2, g) \( \sqrt{254.45 \cdot \left(\text{s1}_{\text{max}} \cdot \text{dmax}_1 + \text{s2}_{\text{max}} \cdot \text{dmax}_2\right) + g \cdot \left(\text{s1}_{\text{max}} + \text{s2}_{\text{max}}\right)} + \text{Ve_max} \cdot \text{Ve_max} \)

#define Vs_max5(Ve_max, s1_max, dmax_1, dmax_2, g, wv, wt) \( \sqrt{127.23 \cdot s1_{\text{max}} \cdot \left(\text{dmax}_1 + \text{dmax}_2\right) + 2 \cdot g} + \text{Ve_max} \cdot \text{Ve_max} \)

#define Vs_max6(Ve_max, s1_max, dmax_1, dmax_2, g) \( \sqrt{127.23 \cdot s1_{\text{max}} \cdot \left(\text{dmax}_1 + \text{dmax}_2\right) + 2 \cdot g} + \text{Ve_max} \cdot \text{Ve_max} \)

#define Vs_max7(Ve_max, s1_max, s2_max, s3_max, dmax_1, dmax_2, dmax_3, g, wv, wt) \( \sqrt{254.45 \cdot \left(\frac{wv}{wv + wt}\right) \cdot \left(\text{sl}_{\text{max}} \cdot \text{dmax}_1 + \text{s2}_{\text{max}} \cdot \text{dmax}_2 + \text{s3}_{\text{max}} \cdot \text{dmax}_3\right) + g \cdot \left(\text{s1}_{\text{max}} + \text{s2}_{\text{max}} + \text{s3}_{\text{max}}\right)} + \text{Ve_max} \cdot \text{Ve_max} \)

#define Vs_max8(Ve_max, s1_max, s2_max, s3_max, dmax_1, dmax_2, dmax_3, g) \( \sqrt{254.45 \cdot \left(\text{s1}_{\text{max}} \cdot \text{dmax}_1 + \text{s2}_{\text{max}} \cdot \text{dmax}_2 + \text{s3}_{\text{max}} \cdot \text{dmax}_3\right) + g \cdot \left(\text{s1}_{\text{max}} + \text{s2}_{\text{max}} + \text{s3}_{\text{max}}\right)} + \text{Ve_max} \cdot \text{Ve_max} \)

#define Vs_max9(Ve_max, s1_max, dmax_1, dmax_2, dmax_3, g, wv, wt) \( \sqrt{84.82 \cdot s1_{\text{max}} \cdot \left(\text{dmax}_1 + \text{dmax}_2 + \text{dmax}_3\right) \cdot \left(\frac{wv}{wv + wt}\right) + 3 \cdot g} + \text{Ve_max} \cdot \text{Ve_max} \)

#define Vs_max10(Ve_max, s1_max, dmax_1, dmax_2, dmax_3, g) \( \sqrt{84.82 \cdot s1_{\text{max}} \cdot \left(\text{dmax}_1 + \text{dmax}_2 + \text{dmax}_3\right) + g} + \text{Ve_max} \cdot \text{Ve_max} \)

#define Vs_max11(Ve_max, s1_max, dmax_1, dmax_2, dmax_3, g, wv, wt) \( \sqrt{254.45 \cdot s1_{\text{max}} \cdot \left(\frac{wv}{wv + wt}\right) \cdot \text{dmax} + g} + \text{Ve_max} \cdot \text{Ve_max} \)

#define Vs_max12(Ve_max, s1_max, dmax_1, dmax_2, dmax_3, g) \( \sqrt{254.45 \cdot s1_{\text{max}} \cdot \text{dmax} + g} + \text{Ve_max} \cdot \text{Ve_max} \)

#define Vi_max(Vs_max, Ve_max) \( \sqrt{(\text{Vs_max} \cdot \text{Vs_max}) + (\text{Vs_max} \cdot \text{Ve_max})} \)

#define Ve_max(Vs_max, Vi_max) \( \sqrt{(\text{Vi_max} \cdot \text{Vi_max}) - (\text{Vs_max} \cdot \text{Vs_max})} \)

#define t_max(Vs_max, Ve_max, dmax, g) \( (\text{Vs_max} - \text{Ve_max}) / (35.34 \cdot \text{dmax} + g) \)

#define s1_max1(Vs_max, Ve_max, dmax, g, wv, wt) \( ((\text{Vs_max} \cdot \text{Vs_max}) - (\text{Ve_max} \cdot \text{Ve_max})) / (254.45 \cdot (\text{wv}(\text{wv} + \text{wt}) \cdot \text{dmax} + g)) \)

#define s1_max2(Vs_max, Ve_max, dmax, g) \( ((\text{Vs_max} \cdot \text{Vs_max}) - (\text{Ve_max} \cdot \text{Ve_max})) / (254.45 \cdot (\text{dmax} + g)) \)

#define fmax_1(Vs_max, Ve_max, s1_max, g, wv, wt)
```c
#define fmax_2(Vs_max, Ve_max, sl_max, g) ((Vs_max*Vs_max)-(Ve_max*Ve_max)+(254.45*sl_max*g)/(254.45*sl_max*(wv/(wv+wt))))

#define pd_max(Vs_max, pt_max) (0.278*Vs_max*pt_max)

#define ts_max(Vs_max, pt_max, dmax, g) (0.278*Vs_max*pt_max) + (Vs_max*Vs_max)/(254.45*(dmax+g))

#define fl(vs, s) (vs*vs)/(254.45*s)
#define f2(vs, t) vs/(35.34*t)
#define f3(s, t) 0.204*s/(t*t)
#define s1(vs, f) vs*vs/(254.45*f)
#define s2(vs, t) 0.139*vs*t
#define s3(f, t) 4.91*f*(t*t)
#define t1(vs, f) vs/(35.34*f)
#define t2(s, f) 0.452*sqrt(s/f)
#define t3(vs, s) 7.20*s/vs
#define vs1(s, f) 15.9*sqrt(s*f)
#define vs2(s, t) 7.20*s/t
#define vs3(f, t) 35.34*f*t
```
TITLE.C

#include <stdio.h>
#include <conio.h>
#include <graphics.h>
#include <stdlib.h>

void title(void);
void page_one(void);
void press_return(void);
void auto_initialization(void);

main()
{
    page_one();
    title();
}

void title(void)
{
    textbackground(LIGHTGRAY);
    clrscr();
    textcolor(BLUE);
    gotoxy(45,3); cprintf("SKIDPRO");
    textbackground(LIGHTCYAN);
    textcolor(BLACK);
    gotoxy(7,5); cprintf(" ");
    gotoxy(7,6); cprintf(" A Program to Determine Speed, Skidding Time, and ");
    gotoxy(7,7); cprintf(" Skidding Distances of Vehicles. ");
    gotoxy(7,8); cprintf(" Implemented in Turbo C for MS-DOS V 5.0 or newer. ");
    gotoxy(7,9); cprintf(" ");
    gotoxy(7,10); cprintf(" Developed by Abdullah S. Nuruddin as Thesis for ");
    gotoxy(7,11); cprintf(" the degree of Master of Science ");
    gotoxy(7,12); cprintf(" ");
    gotoxy(7,13); cprintf(" Thesis/Program Director ");
    gotoxy(7,14); cprintf(" Helmut T. Zwahlen, Ph.D. ");
    gotoxy(7,15); cprintf(" ");
    gotoxy(7,16); cprintf(" Copyr. 1993 H.T. Zwahlen Ph.D, A.S. Nuruddin, Ohio Univ. ");
    gotoxy(7,17); cprintf(" Department of Industrial and Systems Engineering ");
    gotoxy(7,18); cprintf(" Ohio University, Athens, Ohio 45701-2979, U.S.A. ");
    gotoxy(7,19); cprintf(" November 1993 ");
    gotoxy(7,20); cprintf(" ");
    textbackground(LIGHTGRAY);
    press_return();
}
void page_one(void)
{
    auto_initialization();
    setgraphmode(0);
    setbkcolor(LIGHTGRAY);
    setcolor(LIGHTBLUE);
    settextstyle(0, 0, 5);
    outtextxy(130, 40, "WELCOME TO");
    outtextxy(145, 100, "SKIDPRO");
    setcolor(MAGENTA);
    settextstyle(0, 0, 1);
    outtextxy(225, 180, "press any key to continue");
    getch();
    closegraph();
}
APPENDIX B

COMPLETED QUESTIONNAIRE FORMS OF USERS FEEDBACK
QUESTIONNAIRE DESIGNED FOR USER FEEDBACK AFTER USING SKIDPRO

Date: 27/11/53

Name:

Address:

City/Town  State  Zip

Educational background  [B.S.  Computer (Electrical - Physics)]

Company (if applicable)

1. What problem(s)(if any) did you encounter working with the Software (SKIDPRO)?.

2. Were the menu options self-explanatory relative to what you were looking to find?.

3. Did the screen display help in terms of readability and information digestion?.

4. Are the questions you encountered in the interactive session structured to your understanding?.
5. What is the thing(s) you like or dislike about SKIDPRO?

I like the menu because it makes you directed to the bottom.

6. What is your overall assessment of the software?

I think it's... excellent work. I'll give it A grade.

7. Is the interactive process clear/vague/in-between?

Clear

8. What problem(s) did you see in using the traditional method?

Too many calculations and time consuming.

9. If formulas were not provided, would you have been able to solve the problems using the traditional method?

No, but it will take much time.

10. Which approach would you be comfortable with, the traditional way or using the Software Package, and why?

I prefer the software because it is fast, easy, and fun to work with.
QUESTIONNAIRE DESIGNED FOR USER FEEDBACK AFTER USING SKIDPRO

Date: 2/8/93

Name: Ziad Akriw

Address: Mill St. Apt 9

City/Town Athens

State OH

Zip 45721

Educational background Masters Degree (Electrical Engg)

Company (if applicable)

1. What problem(s)(if any) did you encounter working with the Software (SKIDPRO)?
   In one or two places, the units are mixed.

2. Were the menu options self-explanatory relative to what you were looking to find?
   Yes

3. Did the screen display help in terms of readability and information digestion?
   Yes

4. Are the questions you encountered in the interactive session structured to your understanding?
   Yes
5. What is the thing(s) you like or dislike about SKIDPRO?

I like the colorful menus and graphics.

6. What is your overall assessment of the software?

Very good.

7. Is the interactive process clear/vague/in-between?

In-between.

8. What problem(s) did you see in using the traditional method?

Time taken to do the calculations.

9. If formulas were not provided, would you have been able to solve the problems using the traditional method?

I do not think so.

10. Which approach would you be comfortable with, the traditional way or using the Software Package, and why?

The software. It is fast and simple and you do not have to worry about whether you know the formulas or not.
QUESTIONNAIRE DESIGNED FOR USER FEEDBACK AFTER USING SKIDPRO

Date: 21st Aug. 1993
Name: Adel
Address: 13 Stewart Street
City/Town: Athens  State: OH  Zip: 45701
Educational background: Ph.D Student in Physics
Company (if applicable) —

1. What problem(s)(if any) did you encounter working with the Software (SKIDPRO)?
   — None

2. Were the menu options self-explanatory relative to what you were looking to find?.
   — Yes

3. Did the screen display help in terms of readability and information digestion?.
   — Yes

4. Are the questions you encountered in the interactive session structured to your understanding?.
   — Yes
5. What is the thing(s) you like or dislike about SKIDPRO?
   
   "It is very easy to run. Also screen display is very good."

6. What is your overall assessment of the software?
   
   B+

7. Is the interactive process clear/vague/in-between?
   
   Clear

8. What problem(s) did you see in using the traditional method?
   
   Calculations too cumbersome and subject to mistakes.

9. If formulas were not provided, would you have been able to solve the problems using the traditional method?
   
   Yes

10. Which approach would you be comfortable with, the traditional way or using the Software Package, and why?
    
    The software because it makes the calculation simple and fast. It also gives more information that you even expect. As calculation is also quick.
QUESTIONNAIRE DESIGNED FOR USER FEEDBACK AFTER USING
SKIDPRO

Date: 27/11/93

Name: Mohammed Shali
t

Address: 186-22 Wolfe street

City/Town Athens State OH Zip 45701

Educational background NC Electrical Engs

Company (if applicable)

1. What problem(s)(if any) did you encounter working with the Software
(SKIDPRO)?

2. Were the menu options self-explanatory relative to what you were looking to
find?.

3. Did the screen display help in terms of readability and information digestion?.

4. Are the questions you encountered in the interactive session structured to your
understanding?.
5. What is the thing(s) you like or dislike about SKIDPRO?

6. What is your overall assessment of the software?

7. Is the interactive process clear/vague/in-between?

8. What problem(s) did you see in using the traditional method?

9. If formulas were not provided, would you have been able to solve the problems using the traditional method?

10. Which approach would you be comfortable with, the traditional way or using the Software Package, and why?
QUESTIONNAIRE DESIGNED FOR USER FEEDBACK AFTER USING
SKIDPRO

Date: 15/11/93

Name: HAITHAM ALKHATIB

Address: 186-C2 WOLFE STREET

City/Town: ATHENS State: OH Zip: 45701

Educational background: PH.D CANDIDATE - MATHEMATICS

Company (if applicable)

1. What problem(s)(if any) did you encounter working with the Software (SKIDPRO)?

2. Were the menu options self-explanatory relative to what you were looking to find?.

   Yes, very good

3. Did the screen display help in terms of readability and information digestion?.

   Yes

4. Are the questions you encountered in the interactive session structured to your understanding?.

   Yes
5. What is the thing(s) you like or dislike about SKIDPRO?
   *I don't know. I think I like everything I saw*

6. What is your overall assessment of the software?
   *I am not a computer programmer, but I think this is very good work.*

7. Is the interactive process clear/vague/in-between? *Clear*

8. What problem(s) did you see in using the traditional method?
   *Solving using a calculator is still time consuming*

9. If formulas were not provided, would you have been able to solve the problems using the traditional method? *Yes*

10. Which approach would you be comfortable with, the traditional way or using the Software Package, and why? *The Software Package. It gives fast results*
QUESTIONNAIRE DESIGNED FOR USER FEEDBACK AFTER USING
SKIDPRO

Date: 27/10/93

Name: Atef Imam

Address: 13 Stewart Str

City/Town: Athens State: OH Zip: 45201

Educational background: M.S. Industrial & Systems Engg.

Company (if applicable) __________________________

1. What problem(s) (if any) did you encounter working with the Software (SKIDPRO)?
   None

2. Were the menu options self-explanatory relative to what you were looking to find?.
   Yes

3. Did the screen display help in terms of readability and information digestion?.
   Yes

4. Are the questions you encountered in the interactive session structured to your understanding?.
   Yes
5. What is the thing(s) you like or dislike about SKIDPRO?

The menu display and graphics are really nice.

6. What is your overall assessment of the software?

Excellent

7. Is the interactive process clear/vague/in-between?

Very clear

8. What problem(s) did you see in using the traditional method?

Takes too much time and energy

9. If formulas were not provided, would you have been able to solve the problems using the traditional method?

No

10. Which approach would you be comfortable with, the traditional way or using the Software Package, and why?

The software because it's fast, easy and you do not have to worry whether your answers are right (as with the traditional method) since the software gives you the right results.
QUESTIONNAIRE DESIGNED FOR USER FEEDBACK AFTER USING
SKIDPRO

Date: 04/13/93
Name: Sayed Abbass

Address: Apt 2 Oak Ridge

City/Town: Albany State: OH Zip: 43701

Educational background: M.S. Mechanical Engineering

Company (if applicable): Ohio Automation

1. What problem(s) (if any) did you encounter working with the Software (SKIDPRO)?

2. Were the menu options self-explanatory relative to what you were looking to find?
   yes, the options are really simple to follow

3. Did the screen display help in terms of readability and information digestion?
   yes, I personally like the displays because they are not crowded with information.

4. Are the questions you encountered in the interactive session structured to your understanding?
   yes, they look simple and straightforward, I believe anyone can run this program.
5. What is the thing(s) you like or dislike about SKIDPRO?
   The menus and graphics are well designed

6. What is your overall assessment of the software?
   To me, it's very good

7. Is the interactive process clear/vague/in-between? Clear

8. What problem(s) did you see in using the traditional method?
   Having to understand the formulas and do some tedious calculations

9. If formulas were not provided, would you have been able to solve the problems using the traditional method? I am not sure, I would have been a

10. Which approach would you be comfortable with, the traditional way or using the Software Package, and why? Software, simple, fast and fun to run.
APPENDIX C

INDEX OF SYMBOLS USED IN THE PROGRAM
APPENDIX C

INDEX OF SOME OF THE SYMBOLS USED IN PROGRAM

d_{max}  Maximum resultant or combined drag factor generated by vehicle
        tires (dimensionless).

d_{max 1} Maximum resultant or combined drag factor generated by vehicle tires
        on road surface type 1 (dimensionless).

d_{max 2} Maximum resultant or combined drag factor generated by vehicle tires
        on road surface type 2 (dimensionless).

d_{max 3} Maximum resultant or combined drag factor generated by vehicle tires
        on road surface type 3 (dimensionless).

d_{b}     Braking distance. i.e. the distance from the application of the brakes
        till the point the brakes are released, feet or meters.

d_{p}     Driver perception-reaction time. The time it takes to perceive the need to
        react to the time of reaction, seconds.

End_{max} maximum speed at the end of braking, mph or km/h.

f         Coefficient of friction of a road surface (dimensionless).

F         Frictional force acting between the tires and the road surface,
        measured in pounds.

max       maximum value for a graph function

max friction maximum coefficient of friction (dimensionless).
max_pd  function of maximum perception-reaction distance (feet or meters).
max_stop maximum braking distance (feet or meters).
max_total maximum total stopping distance (feet or meters).
min minimum value for a graph function
min_pd function of minimum perception-reaction distance (feet or meters).
pd_max function of the maximum perception-reaction distance (feet or meters)
pt_max maximum perception-reaction time (seconds).
pt_min minimum perception-reaction time (seconds).
rd-max Maximum rear axle drag factor of vehicle (dimensionless).
f-maxx Maximum front axle drag factor of vehicle (dimensionless).
fmax1 Maximum coefficient of friction of road surface 1 (dimensionless).
fmax2 Maximum coefficient of friction of road surface 2 (dimensionless).
fmax3 Maximum coefficient of friction of road surface 3 (dimensionless).
f_r Coefficient of friction on the two front wheels (dimensionless).
f_r Coefficient of friction on the two rear wheels (dimensionless).
g Gravitational force represented as 32.2 ft/sec^2 or 9.81 meters/sec^2.
g_r Grade percent of road. i.e. the steepness of a road (dimensionless).
h Height to which an axle of a vehicle is lifted, feet or meters.
L Length of the wheelbase of a vehicle. It measures from the center of the front wheel to the center of the rear wheel, measured in feet.
m Mass of a vehicle, in pounds/feet/sec^2 (slugs) or newtons/meters/sec^2 (kg).
maxx pointer to the address of the maximum screen length in the X axis
*maxy  pointer to the address of the maximum screen length along the Y axis
R    Radius of vehicle tire with wheels, measured in feet.
s    Sliding or skidding distance of a vehicle, feet.
size  scaling factor on the axis.
s1-max  Maximum sliding or skidding distance of vehicle on road surface type
        1, feet or meters.
s2-max  Maximum sliding or skidding distance of vehicle on road surface type
        2, feet or meters.
s3-max  Maximum sliding or skidding distance of vehicle on road surface type
        3, feet or meters.
temp  size of scaling on the axis
temp1  minimum value return from each function
temp2  maximum value return from each function
tmax  Maximum skidding time of a vehicle involved in skidding. It is the time
        between the application of the brakes and the vehicle starts skidding
        to the time the vehicle comes to a stop at the end of braking.
TSD  Total stopping distance. This is the distance traveled during the
      perception-reaction time during which the vehicle does not slow down plus
      the distance it takes to bring the vehicle to a stop by braking.
ts_max  function for the maximum total stopping distance (feet, meters)
V_{e-max}  Maximum end or final speed of a vehicle at the end of braking, mph.
V_{t-max}  Maximum initial or beginning speed of vehicle at the instant of brake
applications (mph or km/h).

\( V_{s-\text{max}} \)  Maximum skidding or sliding speed of a vehicle (mph or km/h).

\( W \)  Weight of vehicle (lbs or newtons).

\( W_a \)  Weight of an automobile (lbs or newtons).

\( W_v \)  Weight of a vehicle (lbs or newtons).

\( W_T \)  Weight of trailer (lbs or newtons).

\( w_f \)  Weight on the front axle of a vehicle when the rear end is lifted \( h \) feet (lbs or newtons).

\( w_a \)  Weight on one axle when the other is hoisted \( h \) feet (lbs or newtons).

\( w_r \)  Weight on the rear axle of a vehicle when the front end is lifted \( h \) feet (lbs or newtons).

\( x \)  Height of the center of mass of a vehicle, feet or meters.

\( x_{\text{min}} \)  minimum value on the X axis

\( x_{\text{max}} \)  maximum value on the Y axis

\( \text{XTicks} \)  scale factor division on the X axis

\( \text{YTicks} \)  scale factor division on the Y axis

\( y_f \)  Location of the center of mass of vehicle measured from the front axle (feet or meters).

\( y_r \)  Location of the center of mass of vehicle measured from the rear axle (feet or meters).
APPENDIX D

INDEX OF FUNCTIONS USED IN THE PROGRAM
APPENDIX D

INDEX OF FUNCTIONS

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>auto_initialization</td>
<td>Initialize the graphics mode</td>
</tr>
<tr>
<td>Calculator</td>
<td>This function works like a calculator in that from a set of four parameters, if the user provides values for any combination of two of the parameters, the function will calculate the values for the other two parameters.</td>
</tr>
<tr>
<td>Coeff_Friction</td>
<td>This function determines the coefficient of friction of a road surface on which skidding takes place.</td>
</tr>
<tr>
<td>Combined_Drag</td>
<td>This function is used when the front and rear axles of a vehicle skidding have different drag factors. In such a case a combined drag formula is invoked to determine the combined drag factor for the whole vehicle.</td>
</tr>
<tr>
<td>Drag_Factor1</td>
<td>The function determines the resultant drag factor if a vehicle skids on one road surface with some of the wheels not sliding.</td>
</tr>
<tr>
<td>Drag_Factor2</td>
<td>This function determines the combined drag factor of a vehicle skidding on two different road surfaces in which some of its wheels slides on one surface and the others slides on the other surface.</td>
</tr>
<tr>
<td>Drag_Factor3</td>
<td>This function determines the resultant drag factor of a vehicle sliding on three different road surfaces in which the some wheels</td>
</tr>
</tbody>
</table>
slide do not slid on all the three surfaces.

downhill Indicates that skidding is in a downhill trend
level Indicates that skidding is on a level pavement
uphill Indicates that skidding in uphill
final_speed Segment of program calculating the final (end) speed after braking
initial_speed Segment of program calculating the beginning speed before braking
press_return Displays 'press_return' on the screen for user action
road_structure A menu displaying the different structures (level, downhill, uphill)
road_type Menu displaying the number of pavement types skidding vehicle traveled on.
Skid_Distance This functions calculates the distance a vehicle skids before it comes to a stop or otherwise.
Skid_Time This function determines the time between the start of skidding to the end of skidding.
WrongChoice This function warms the user of an incorrect choice made from a set of options.
Wrong_answer This function also warms the user of an incorrect selection. It is invoked when a selection is not from the list to be selected.
plotgraph  plots the graphs of various functions
my_function  Returns the pixel values required to plot the graph of a given function.
mainmenu  Displays the main menu of the entire program
page_one  Contains information pertaining to title, program developer, copyright.
title  Displays the name 'SKIDPRO' used as the name of the software
setgraph  Draws the X-Y axis, scales and labels them.
yscale  Resets the values for the Y axis
xscale  Resets the values for the X axis
Get_input  Function to guide against unreasonable user inputs or illegal characters. It displays INPUT OUT OF RANGE or ILLEGAL INPUT depending on the type of input made until the right input is made.
Scroll  This function scrolls up the contents on the screen after every 15 lines.
APPENDIX E

DERIVATIONS AND LIST OF FORMULAS USED IN THE PROGRAM
# CONTENTS OF FORMULAS DERIVED

<table>
<thead>
<tr>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>English units - conversion of ft/sec. to miles/hr</td>
<td>E-5</td>
</tr>
<tr>
<td>Initial speed in terms of final speed, distance and drag factor</td>
<td>E-5</td>
</tr>
<tr>
<td>Final speed in terms of initial speed, distance and drag factor</td>
<td>E-7</td>
</tr>
<tr>
<td>Distance in terms of initial speed, final speed and drag factor</td>
<td>E-7</td>
</tr>
<tr>
<td>Initial speed in terms of final speed, time and drag factor</td>
<td>E-8</td>
</tr>
<tr>
<td>Initial speed in terms of time, distance and drag factor</td>
<td>E-9</td>
</tr>
<tr>
<td>Distance in terms of initial speed, time and drag factor</td>
<td>E-11</td>
</tr>
<tr>
<td>Distance in terms of final speed, time and drag factor</td>
<td>E-11</td>
</tr>
<tr>
<td>Time in terms of initial speed, distance and drag factor</td>
<td>E-12</td>
</tr>
<tr>
<td>Time in terms of final speed, distance and drag factor</td>
<td>E-14</td>
</tr>
<tr>
<td>Metric units - Conversion of meters/sec to km/hour</td>
<td>E-16</td>
</tr>
<tr>
<td>Formulas involving drag factor on grades</td>
<td>E-17</td>
</tr>
<tr>
<td>Using the English units for formulas involving grades</td>
<td>E-24</td>
</tr>
<tr>
<td>Using the Metric units for formulas involving grades</td>
<td>E-27</td>
</tr>
<tr>
<td>Formulas involving grades in which final speed is not zero</td>
<td>E-31</td>
</tr>
<tr>
<td>Formulas involving changing deceleration</td>
<td>E-36</td>
</tr>
</tbody>
</table>
DERIVATIONS OF EQUATIONS

The variables describing motion involving acceleration or deceleration are: time, t, distance, s, acceleration, a, initial velocity, v_i, and final velocity, v_e. If any three of these variables are known, the other two can be easily derived.

If a vehicle is changing speed (acceleration or deceleration) steadily, its average velocity \( v_a \) is half the sum of the initial velocity, \( v_i \), plus the final velocity, \( v_e \). Thus

\[
v_a = \frac{(v_i + v_e)}{2}
\]  

(1)

The distance, s, required to stop the vehicle after a certain time, t is that time multiplied by the average velocity, \( v_a \). Then

\[
s = tv_a
\]  

(2)

Substitute \( v_a \) in Equation (2)
From Equation (3)

\[ s = \frac{(v_i + v_f)}{2} \]  

(3)

The time, \( t \), can be derived from Equation (5) as

\[ t = \frac{2s}{(v_i + v_f)} \]  

(4)

Acceleration or deceleration, \( a \), is the rate at which velocity (speed) increases or decreases per unit time, \( t \). That is

\[ a = \frac{(v_f - v_i)}{t} \]  

(5)

When final velocity is greater than initial velocity, then velocity is increasing (accelerating), \( a \) is positive and when final velocity is less than initial velocity, then velocity is decreasing (decelerating), \( a \) is negative. Acceleration or deceleration is measured in feet-per second per second (ft/sec^2), or meters-per second per second (meters/sec^2).

The time, \( t \), can be derived from Equation (5) as
From Equations (4) and (6)

\[ t = \frac{(v_i - v_f)}{a} \]  \hspace{1cm} (6)

Cross multiplying gives

\[ \frac{2s}{(v_i + v_f)} = \frac{(v_i - v_f)}{a} \]

Cross multiplying gives

\[ 2sa = (v_i - v_f)(v_i + v_f) \]

\[ v_i^2 = v_e^2 + 2sa \]

Take the square root of both sides
But acceleration or deceleration, $a$ is the gravitational force, $g$ acting on the vehicle times the horizontal force (drag factor), $f$, acting between the tires and road. That is

$$a = fg$$ (8)

Substituting $a$ in Equation (7)

$$v_i = \sqrt{v_e^2 + 2sa}$$ (7)

$$v_i = \sqrt{v_e^2 + 2sfg}$$ (9)

But the gravitational force, $g = 32.2 \text{ ft/sec}^2$ or 9.81 meters/sec$^2$.

Since velocity is either in ft/sec or meters/sec, it can be converted to miles/hr or km/hr which are normally the units used when speed is used instead of velocity.
English units - Conversion of ft/sec to miles/hour

\[ v \, (ft/sec) = V \times \frac{5280}{3600} \, (miles/hr) \]

\[ v \, (ft/sec) = 1.4667V \, (miles/hr) \]  \hspace{1cm} (10)

where \( v \) is velocity and \( V \) is speed.

Deriving initial speed in terms of final speed, distance and drag factor.

Substitute \( g = 32.2 \, ft/sec^2 \) and using the conversion of Equation (10), Equation (9) becomes

\[ 1.4667V_i = \sqrt{V_f^2(1.4667^2) + 2sf(32.2)} \]

Square both sides
\[ 1.4667^2 v_i^2 = 1.4667^2 v_e^2 + 64.4sf \]

Divide both sides by 1.4667^2

\[ v_i^2 = v_e^2 + \frac{64.4sf}{1.4667^2} \]

Simplify the term on the left

\[ v_i^2 = v_e^2 + 30sf \quad (11) \]

Take the square root of both sides

\[ v_i = \sqrt{v_e^2 + 30sf} \quad (12) \]
Deriving final speed in terms of initial speed, distance and drag factor.

Equation 11 can be written in terms of $V_e$ as

$$V_e^2 = V_i^2 - 30sf$$  \hspace{1cm} (13)

Taking square root on both sides,

$$V_e = \sqrt{V_i^2 - 30sf}$$  \hspace{1cm} (14)

Deriving distance in terms of initial speed, final speed and drag factor.

Rewriting Equation 11 so that $s$ appears on the left side gives

$$30sf = V_i^2 - V_e^2$$  \hspace{1cm} (15)

Divide both sides by $30f$ gives
Also dividing both sides of Equation 15 by 30s gives

$$s = \frac{v_i^2 - v_e^2}{30f}$$  \hspace{1cm} (16)

$$f = \frac{v_i^2 - v_e^2}{30s}$$  \hspace{1cm} (17)

When vehicle slides to a stop, the final speed $v_e = 0$. Also all the above formulas are only valid for level surfaces (grade is zero).

**Deriving initial speed in terms of final speed, time and drag factor.**

Cross multiply Equation 5 or 6,

$$at = v_i - v_e$$  \hspace{1cm} (18)

Using the conversion factor of Equation 10, and $a=32.2f$, Equation 18 becomes
\[ 32.2\text{ft} = 1.4667(V_i - V_e) \quad (19) \]

Divide both sides by 32.2 an rewriting Equation 19

\[ 0.0455V_i = 0.0455V_e + \text{ft} \quad (20) \]

Divide both sides by 0.0455,

\[ V_i = V_e + 22.0\text{ft} \quad (21) \]

Deriving initial speed in terms of time, distance and drag factor.

From Equation 5,

\[ v_e = v_i - at \quad (22) \]

Substitute Equation 22 in Equation 3
Simplifying Equation 23 gives

\[ s = \frac{t(v_i + v_i - at)}{2} \]  \hspace{1cm} (23)

From Equation 24

\[ s = v_i t - \frac{at^2}{2} \]  \hspace{1cm} (24)

Applying the conversion in Equation 10 to Equation 25 and g=32.2f gives

\[ 1.4667v_i = s + \frac{32.2f t^2}{2} \]  \hspace{1cm} (25)

Dividing both sides of Equation 26 by 1.4667t gives
Simplifying equation 27 gives

\[ V_i = \frac{s}{1.4667t} + \frac{32.2ft^2}{2 \times 1.4667t} \]  

(27)

\[ V_i = 0.682\frac{s}{t} + 10.98ft \]  

(28)

Deriving distance in terms of initial speed, time and drag factor.

Applying the conversion in Equation 10 to Equation 25, and using \( a = 32.2f \),

\[ s = 1.4667V_i^t - 16.1ft^2 \]  

(29)

Deriving distance in terms of final speed, time and drag factor.

From Equation 22,
\[ v_i = v_e + at \]  

Substitute Equation 30 in Equation 3

\[ s = t \frac{(v_e + v_e + at)}{2} \]  

Simplifying gives

\[ s = v_e t + \frac{at^2}{2} \]  

Applying the conversion in Equation 10 to Equation 32,

\[ s = 1.4667V_e t + 16.1ft^2 \]  

Deriving time in terms of initial speed, distance and drag factor.
Taking all terms in Equation 25 containing \( t \) to left side.

\[
\nu f - \frac{at^2}{2} = s
\]  

(34)

Multiply both sides of the equation by \( 2/a \) and rearrange:

\[
\frac{2\nu f}{a} - t^2 = \frac{2s}{a}
\]  

(35)

Multiplying both sides by -1 and adding the term \( \frac{v_i^2}{a^2} \) to both sides:

\[
t^2 - \frac{2\nu f}{a} + \left(\frac{v_i}{a}\right)^2 = \left(\frac{v_i}{a}\right)^2 - \frac{2s}{a}
\]  

(36)

The three terms on the left side is the square of the term \( t - \nu/a \). Hence taking the square of both sides of equation 36 gives:

\[
t - \frac{\nu i}{a} = \sqrt{\frac{v_i^2}{a^2} - \frac{2s}{a}}
\]  

(37)
Applying the conversion in Equation 10 to Equation 38 gives:

\[
t = \sqrt{\frac{\nu_i^2}{a^2} - \frac{2s}{a} + \frac{\nu_i}{a}}
\]

(38)

Simplifying the equation gives:

\[
t = \sqrt{1.4667^2 \nu_i^2 - \frac{2s}{32.2f} + \frac{1.4667\nu_i}{32.2f}}
\]

(39)

Deriving time in terms of final speed, distance and drag factor.

Multiply both sides of Equation 32 by 2/a and rearrange to have the t's on the left side:

\[
t^2 + \frac{2\nu_f t}{a} = \frac{2s}{a}
\]

(41)
Add the term \((v_e/a)^2\) to both sides of the equation to complete the square on the left side:

\[
t^2 + \frac{2v_e t}{a} + \left(\frac{v_e}{a}\right)^2 = \frac{2s}{a} + \left(\frac{v_e}{a}\right)^2
\]  

(42)

The left side represents the square of \((t+v_e/a)\), hence taking the square root of Equation 42 gives:

\[
t + \frac{v_e}{a} = \sqrt{\frac{v_e^2}{a^2} + \frac{2s}{a}}
\]  

(43)

Taking \((v_e/a)\) to the right side of the equation gives:

\[
t = \sqrt{\frac{v_e^2}{a^2} + \frac{2s}{a}} - \frac{v_e}{a}
\]  

(44)

Applying the conversion in Equation 10 to Equation 44 gives:

\[
t = \sqrt{\frac{1.4667^2 \cdot \frac{V_e^2}{32.2f^2}}{32.2f} + \frac{2s}{32.2f} - 1.4667 \cdot \frac{V_e}{32.2f}}
\]  

(45)
Simplifying the equation gives:

\[ t = \sqrt{\frac{0.00208 V_e^2}{f^2} + 0.0621 \frac{s}{f}} - 0.0455 \frac{V_e}{f} \]  

(46)

Metric units - Conversion of meters/sec. to kilometers per hour

\[ v \text{ (meters/sec)} = V \times \frac{1000}{3600} \text{ (km/hr)} \]

(47)

Using the metric units conversion and \( a = 9.81f \), the derivations above can be converted into metric units. Also all the formulas are true for zero grades and constant deceleration.
DRAG FACTOR ON GRADES

When the road is not level, a downgrade (downhill skidding) makes stopping distance greater whether all its wheels are locked or not, and an upgrade (uphill skidding) makes stopping distance less. That is to say a grade, n, increases or decreases the drag factor rather than the coefficient of friction. A grade by definition is a measure of the rise or fall, h, in the horizontal distance, d. Thus in equation form

\[
\frac{n}{d} = \frac{h}{d} \tag{48}
\]

\[
h = dn \tag{49}
\]

\[
d = \frac{h}{n} \tag{50}
\]

Figure E-1 shows the relationships involved in computing the drag factor on a grade.
Figure E-1  Relationships involved in equation for drag factor on a grade.

Applying the Pythagorean theorem in Figure E-1.

\[ d_n^2 = d^2 + h^2 \]  \hspace{1cm} (51)

Substitute for \( h \) in Equation (49) in Equation (51)

\[ d_n^2 = d^2 + d^2 n^2 \]  \hspace{1cm} (52)

Collect terms containing \( d^2 \)

\[ d_n^2 = d^2 (1 + n^2) \]  \hspace{1cm} (53)

Take square root of both sides

\[ d_n = d \sqrt{1 + n^2} \]  \hspace{1cm} (54)

and

\[ d = \frac{d_n}{\sqrt{1 + n^2}} \]  \hspace{1cm} (55)
From Figure E-1, using geometric analysis of proportionality, the component of the vehicle’s weight \( w_n \) pressing against the sloped surface bears the same relationship to the vehicle’s total weight \( w \) as the horizontal distance, \( d \), bears to the length of the slope \( d_n \). Hence

\[
\frac{w_n}{w} = \frac{d}{d_n}
\]  

(56)

Substitute the value of \( d_n \) in Equation 54

\[
\frac{w_n}{w} = \frac{d}{d\sqrt{1+n^2}}
\]  

(57)

Cancel the \( d \)’s and multiply both sides by \( w \)

\[
w_n = \frac{w}{\sqrt{1+n^2}}
\]  

(58)

The work done \( W \), in slowing to a stop on slope is the work done, \( W_n \), by braking on the surface plus the work done, \( W_h \), in moving the vehicle along the slope. That is
\[ W = W_n + W_h \]  \hspace{1cm} (59)

But the work done \( W_n \) in moving the vehicle along the slope \( d_n \) is given as

\[ W_n = F_n d_n \]  \hspace{1cm} (60)

and the force \( F_n \) parallel to the slope required to move the vehicle along the slope is given as

\[ F_n = f w_n \]  \hspace{1cm} (61)

Substitute \( w_n \) in Equation (58) into Equation (61)

\[ F_n = f \frac{w}{\sqrt{1+n^2}} \]  \hspace{1cm} (62)

Substitute \( F_n \) in Equation (62) into Equation (60)

\[ W_n = F_n d_n \frac{w}{\sqrt{1+n^2}} \]  \hspace{1cm} (63)
Also the work $W_h$ required to lift the vehicle a vertical distance, $h$, while moving a distance, $d_n$, is the vertical force representing the full weight of the vehicle, $w$, times the vertical distance, $h$.

$$W_h = wh$$ \hspace{1cm} (64)

Substitute $h$ in equation (49) into Equation (64)

$$W_h = wdn$$ \hspace{1cm} (65)

Substitute $d$ in Equation (55) into Equation (65)

$$W_h = wn - \frac{d_n}{\sqrt{1+n^2}}$$ \hspace{1cm} (66)

Substitute the values obtained for $W_n$ and $W_h$ in Equation 59

$$W = fd_n \frac{w}{\sqrt{1+n^2}} + d_n n \frac{w}{\sqrt{1+n^2}}$$ \hspace{1cm} (67)
Collect the terms containing

\[ d_n \frac{w}{\sqrt{1+n^2}} \]

Then Equation (67) becomes

\[ W = \frac{w}{\sqrt{1+n^2}} d_n(f+n) \]  \hspace{1cm} (68)

But the energy, \( W \), of a body in motion is equal to half its mass, \( m \), times the square if its velocity. That is

\[ W = \frac{mv^2}{2} \] \hspace{1cm} (69)

and mass, \( m \), is weight divided by acceleration due to gravity, \( g \). Hence, Equation (69) becomes

\[ W = \frac{wv^2}{2g} \] \hspace{1cm} (70)
Equating Equations 68 and 69, and dividing each side by \( w \)

\[
\frac{v^2}{2g} = \frac{d_n(f+n)}{\sqrt{1+n^2}}
\]  

(71)

Deriving the formulas in the English units:

Substitute \( g = 32.2 \text{ ft/sec}^2 \) and Equation 10 in Equation 71

\[
V^2 \left( \frac{1.4667^2}{2 \times 32.2} \right) = \frac{d_n(f+n)}{\sqrt{1+n^2}}
\]  

(72)

\[
V^2(0.0334) = \frac{d_n(f+n)}{\sqrt{1+n^2}}
\]  

(73)

Divide both sides by 0.0334

\[
V^2 = \frac{d_n(f+n)}{0.0334\sqrt{1+n^2}}
\]  

(74)
With $1/0.0334 = 29.9 = 30$, take the square root of both sides

$$V = \sqrt{\frac{30d_n(f+n)}{\sqrt{1+n^2}}} \quad (75)$$

For all practical purposes, the factor $(1+n^2)^{1/2}$ is insignificant and may be neglected. Hence

$$V = \sqrt{30d_n(f+n)} \quad (76)$$

Replacing $d_n = s$, and $n = g$, Equation (76) becomes

$$V = \sqrt{30s(f+g)} \quad uphill \quad (77)$$

$$V = \sqrt{30s(f-g)} \quad downhill \quad (50)$$

To derive $s$ and $f$, take the square of both sides of Equation 77

$$V^2 = 30s(f+g) \quad (79)$$
Divide both sides by $30(f+g)$

\[
s = \frac{v^2}{30(f+g)} \quad \text{uphill} \quad (80)
\]

\[
s = \frac{v^2}{30(f-g)} \quad \text{downhill} \quad (81)
\]

Cross multiply the terms in Equation (80) or (81)

\[
v^2 = 30sf + 30sg \quad (82)
\]

Take the term containing $f$ to the right side.

\[
30sf = v^2 - 30sg \quad (83)
\]

\[
f = \frac{v^2 - 30sg}{30s} \quad \text{uphill} \quad (84)
\]
where

\[ V = \text{speed in miles per hour} \]
\[ s = \text{distance in feet} \]
\[ f = \text{coefficient of friction} \]
\[ g = \text{grade of road (g is -ive when downhill and +ive when uphill)} \]

Deriving the formulas in the Metric Units:

Substituting \( g = 9.81 \text{ meters/sec}^2 \) and using the conversion in Equation (17), Equation (46) becomes

\[
\frac{0.2778 V^2}{2 \times 9.81} = \frac{d_n(f+n)}{\sqrt{1+n^2}}
\]

\[
V^2(0.0039) = \frac{d_n(f+n)}{\sqrt{1+n^2}}
\]

Divide both sides by 0.0039
\[ V^2 = \frac{d_n(f+n)}{0.0039\sqrt{1+n^2}} \]

Simplify \(1/0.0039 = 254.23\) and take the square root of both sides

\[ V = \frac{254.23d_n(f+n)}{\sqrt{1+n^2}} \]

Again for all practical purposes, the factor \((1+n^2)^{1/2}\) is insignificant and may be neglected.

\[ V = \sqrt{254.23d_n(f+n)} \quad (89) \]

Replacing \(d_n = s\), and \(n = g\), Equation (89) becomes

\[ V = \sqrt{254.23s(f+g)} \quad \text{uphill} \quad (90) \]
To derive $s$ and $f$, square both sides of Equation (90)

$$v^2 = 254.23s(f+g)$$ \hfill (92)

Divide both sides of Equation (92) by $254.23(f+g)$

$$s = \frac{v^2}{254.23(f+g)} \quad \text{uphill} \hfill (93)$$

$$s = \frac{v^2}{254.23(f-g)} \quad \text{downhill} \hfill (94)$$

To find $f$, expand to remove the brackets in Equation 92

$$v^2 = 254.23sf + 254.23sg$$

Take the term containing $f$ to one side of the equation
\[ 254.23sf = V^2 - 254.23sg \]

Divide both sides by 254.23s

\[ f = \frac{V^2 - 254.23sg}{254.23s} \quad \text{uphill} \quad \text{(95)} \]

\[ f = \frac{V + 254.23sg}{254.23s} \quad \text{downhill} \quad \text{(96)} \]

Where

- \( V \) = speed in kilometers per hour
- \( s \) = distance in meters
- \( f \) = coefficient of friction
- \( g \) = grade of road (\( g \) is -ive when downhill and +ive when uphill)
Skidding on grades in which final speed is not zero.

If the speed at end of slowing is not equal to zero, then the speed $V_e$ at the end of slowing is determined by the energy $W_e$ remaining at that time. This energy is expressed as

$$W_e = \frac{mV_e^2}{2} \quad (94)$$

Likewise, the energy $W_s$ required to slow at a speed $V_s$ to a stop can be expressed as

$$W_s = \frac{mV_s^2}{2} \quad (98)$$

The combined energy at the beginning of slowing $W_i$ is the sum of the energy required to slow to a specified distance and the energy remaining at the end of slowing. Hence

$$W_i = W_s + W_e \quad (99)$$

But the combined energy, $W_i$ can also be expressed as
Substitute \( W_i, W_s, \) and \( W_e \) in Equation (99)

\[
\frac{mV_i^2}{2} = \frac{mV_s^2}{2} + \frac{mV_e^2}{2}
\]

Divide both sides of Equation (100) by \( 2/m \)

\[
V_i^2 = V_s^2 + V_e^2
\]

Take the square root of both sides of Equation (101)

\[
V_i = \sqrt{V_s^2 + V_e^2}
\]

\[
V_s = \sqrt{V_i^2 - V_e^2}
\]
Hence, when final speed is not zero, replace $V=V_e$ in Equation (77) and substitute Equation (103) into it.

$$\sqrt{V_i^2-V_e^2} = \sqrt{30s(f+g)}$$

Take the square of both sides

$$V_i^2-V_e^2 = 30s(f+g)$$

$$V_i^2 = 30s(f+g)+V_e^2$$

Take the square root of both sides

$$V_i = \sqrt{30s(f+g)+V_e^2} \quad \text{uphill} \quad (104)$$

$$V_i = \sqrt{V_e^2 + 30s(f-g)} \quad \text{downhill} \quad (105)$$
Equations (80) and (81) becomes

\[
 s = \frac{V_i^2 - V_e^2}{30s(f+g)} \quad \text{uphill} \quad (106)
\]

\[
 s = \frac{V_i^2 - V_e^2}{30(f-g)} \quad \text{downhill} \quad (107)
\]

Equations (84) and (85) becomes

\[
 f = \frac{V_i^2 - V_e^2 - 30sg}{30s} \quad \text{uphill} \quad (108)
\]

\[
 f = \frac{V_i^2 - V_e^2 + 30sg}{30s} \quad \text{downhill} \quad (109)
\]

Likewise, in the metric units, Equations (90) and (91) becomes

\[
 V_i = \sqrt{254.23s(f+g) + V_e^2} \quad \text{uphill} \quad (110)
\]
\[ V_i = \sqrt{254.23s(f-g)+V_e^2} \]  
\text{downhill} \quad (111)

Equations 93 and 94 becomes

\[ s = \frac{V_i^2-V_e^2}{254.23(f+g)} \]  
\text{uphill} \quad (112)

\[ s = \frac{V_i^2-V_e^2}{254.23(f-g)} \]  
\text{downhill} \quad (113)

Equations (95) and (96) becomes

\[ f = \frac{V_i^2-V_e^2-254.23sg}{254.23s} \]  
\text{uphill} \quad (114)

\[ f = \frac{V_i^2-V_e^2+254.23sg}{254.23s} \]  
\text{downhill} \quad (115)
When deceleration is not constant.

Assuming the acceleration or deceleration is not constant, the instantaneous velocity, \( v \), can be found by taking the derivative of the distance, \( s \), with respect to time. Hence

\[
v = \frac{ds}{dt}
\]  

(116)

The instantaneous acceleration can be found by taking the derivative if the velocity, \( v \), with respect to time. Hence

\[
a = \frac{dv}{dt}
\]  

(117)

From Equation (5)

\[
v_e = v_i + at
\]  

(118)

The distance, \( s \), traveled between the time interval \( t_1 \) and \( t_2 \) can be obtained by taking the integral of the final velocity, \( v_e \), with respect to time. Thus
Substitute the value of Equation (118) in Equation (119)

\[ s = \int_{t_1}^{t_2} v_e \, dt \quad (119) \]

Substitute the value of Equation (8) in Equation (120)

\[ s = \int_{t_1}^{t_2} (v_i + at) \, dt \quad (87) \]

Using the conversion in Equation (10) and \( g = 32.2 \text{ ft/sec}^2 \), for the English units, Equation (121) becomes

\[ s = \int_{t_1}^{t_2} (1.4667v_i + 32.2ft) \, dt \quad (122) \]
Substituting Equation (17) and \( g = 9.81 \text{ meters/sec}^2 \), Equation (121) becomes

\[
\frac{\bar{t}_2}{t_1} s = \int_{0}^{\bar{t}_2} \left( 0.2778 V_t + 9.81 ft \right) dt
\]

(123)

Assuming the coefficient of friction is not uniform but changes with skidding time, Equation (122) becomes

\[
\frac{\bar{t}_2}{t_1} s = 1.4667 \int_{t_1}^{\bar{t}_2} V_t dt + 32.2 \int_{t_1}^{\bar{t}_2} f(t) dt
\]

(124)

and Equation (123) becomes

\[
\frac{\bar{t}_2}{t_1} s = 0.2778 \int_{t_1}^{\bar{t}_2} V_t dt + 9.81 \int_{t_1}^{\bar{t}_2} f(t) dt
\]

(125)

Substitute Equation (8) in Equation (5)
Divide both sides by \( g \)

\[
f_g = \frac{v_e - v_i}{t}
\]

(126)

Using \( g = 32.2 \text{ ft/sec}^2 \), Equation (94) can represent the function \( f(t) \) in Equation (92).

Hence Equation (125) will become

\[
s = 1.4667 \int_{t_1}^{t_2} V_i \, dt + 32.2 \int_{t_1}^{t_2} \left( \frac{V_e - V_i}{32.2t} \right) \, dt
\]

(127)

\[
s = 1.4667 \int_{t_1}^{t_2} V_i \, dt + \int_{t_1}^{t_2} \left( \frac{V_e - V_i}{t} \right) \, dt
\]

(128)
Likewise Equation (126) will become

\[ s = 0.2778 \int_{t_1}^{t_2} V_t \, dt + \int_{t_1}^{t_2} \left( \frac{V_e - V_t}{t} \right) \, dt \]  

(129)

Substitute Equation (118) in Equation (3)

\[ s = t \left( \frac{v_i + v_e + at}{2} \right) \]

Removing the brackets and dividing the terms on the left side by 2

\[ s = v \frac{at^2}{2} \]  

(130)

The velocity at any instant can be obtained by taking the integral of \( s \), in Equation (130) with respect to time. Hence the velocity during the time interval \( t_1 \) and \( t_2 \) is
Substitute Equation (130) in Equation (131)

\[ v = \int_{t_1}^{t_2} s \, dt \]  

Substitute Equation (8) in Equation (132)

\[ v = \int_{t_1}^{t_2} (v t + at^2) \, dt \]  

Again using the conversion of Equation (10) and \( g = 32.2 \text{ ft/sec}^2 \), Equation (133) becomes
Using Equation (17) and \( g = 9.81 \text{ meters/sec}^2 \) for the Metric units, Equation (133) becomes

\[
V = \int_{t_1}^{t_2} (1.4667V_t + \frac{1}{2} 32.2 \text{ft}^2) \, dt
\]

Again assuming the coefficient of friction is not uniform but changes with skidding time, Equation (134) will become

\[
V = \int_{t_1}^{t_2} (0.2778V_t + \frac{1}{2} 9.81 \text{ft}^2) \, dt
\]

(135)
\[ V = 1.4667 \int_{t_1}^{t_2} V_t \, dt + 16.1 \int_{t_1}^{t_2} f(t)t^2 \, dt \]  

(136)

and Equation (135) will become

\[ V = 0.2778 \int_{t_1}^{t_2} V_t \, dt + 4.905 \int_{t_1}^{t_2} f(t)t^2 \, dt \]  

(137)

Using Equation (127) to represent the function \( f(t) \), Equation (136) becomes

\[ V = 1.4667 \int_{t_1}^{t_2} V_t \, dt + 16.1 \int_{t_1}^{t_2} \frac{(V_e - V)}{gt} t^2 \, dt \]

Substitute \( g = 32.2 \) ft/sec\(^2\)
Removing the brackets in the second term and grouping $V_i t$'s together, Equation (138) becomes

$$V = [0.9667V_i + 0.5V_j] \int t \, dt$$  \hspace{1cm} (139)$$

Replacing the function $f(t)$ in Equation (137) by Equation (127)

$$V = 0.2778 \int V_i \, dt + 4.905 \int \left( \frac{V_e - V_j}{gt} \right) t^2 \, dt$$  \hspace{1cm} (140)$$

Substitute $g = 9.81$ meters/sec$^2$
Simplify the second term in Equation 141

\[ V = 0.2778 \int_{t_1}^{t_2} V_i \, dt + 4.905 \int_{t_1}^{t_2} \left( \frac{V_e - V_i}{9.81} \right) t^2 \, dt \]  \hspace{1cm} (141)

Taking the integral of Equation (3) with respect to time gives the velocity at any instant. That is

\[ V = \left[ 0.5V_e - 0.22V_i \right] \int_{t_1}^{t_2} t \, dt \]  \hspace{1cm} (142)

Using the conversion in Equation (10), Equation (143) can be written as

\[ V = \frac{1}{2} \left( V_i + V_e \right) \int_{t_1}^{t_2} t \, dt \]  \hspace{1cm} (143)

\[ V_i \text{ and } V_e \text{ are constants and can be taken out of the integral. Hence Equation (144)} \]
\[ V = \int_{t_1}^{t_2} 1.4667 \frac{(V_i + V_e)}{2} \, dt \]  

becomes

\[ V = 0.733(V_i + V_e) \int_{t_1}^{t_2} t \, dt \]  

Using the conversion in Equation (17), Equation (143) becomes

\[ V = \int_{t_1}^{t_2} 0.2778 \frac{(V_i + V_e)}{2} \, dt \]  

or

\[ V = 0.139(V_i + V_e) \int_{t_1}^{t_2} t \, dt \]  

(144)

(145)

(146)

(147)
CONTENTS OF FORMULAS DERIVED

English units - conversion of ft/sec. to miles/hr .................................................. E-5
Initial speed in terms of final speed, distance and drag factor ..................................... E-5
Final speed in terms of initial speed, distance and drag factor .................................... E-7
Distance in terms of initial speed, final speed and drag factor ..................................... E-7
Initial speed in terms of final speed, time and drag factor ......................................... E-8
Initial speed in terms of time, distance and drag factor ............................................. E-9
Distance in terms of initial speed, time and drag factor ............................................. E-11
Distance in terms of final speed, time and drag factor ............................................. E-11
Time in terms of initial speed, distance and drag factor ........................................... E-12
Time in terms of final speed, distance and drag factor ............................................. E-14
Metric units - Conversion of meters/sec to km/hour .................................................. E-16
Formulas involving drag factor on grades .................................................................. E-17
Using the English units for formulas involving grades ................................................. E-24
Using the Metric units for formulas involving grades .................................................. E-27
Formulas involving grades in which final speed is not zero ...................................... E-31
Formulas involving changing deceleration .................................................................. E-36
<table>
<thead>
<tr>
<th>CONTENTS OF USER'S GUIDE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copyright info.</td>
<td>F-2</td>
</tr>
<tr>
<td>Program developer</td>
<td>F-2</td>
</tr>
<tr>
<td>Program director</td>
<td>F-2</td>
</tr>
<tr>
<td>Units option menu</td>
<td>F-3</td>
</tr>
<tr>
<td>Main Menu</td>
<td>F-4</td>
</tr>
<tr>
<td>Speed menu</td>
<td>F-4</td>
</tr>
<tr>
<td>Road surface menu</td>
<td>F-5</td>
</tr>
<tr>
<td>Grade of road menu</td>
<td>F-6</td>
</tr>
</tbody>
</table>
USER'S GUIDE FOR THE SOFTWARE PACKAGE

The software package is designed in an interactive and user-friendly menu driven environment with a units option for the user to work with the program in either the Metric units or the USA units system of measurement. The program runs on MS DOS 5.0 or higher with a EGA or VGA card installation. After making sure this feature exists on the type of computer used, the user simply executes a DOS file called UNIT.BAT by typing UNITS from the drive or floppy disk prompt carrying the UNIT.BAT file and pressing the <ENTER> key. The first screen to be displayed is the one below. This display welcomes you to the software. The name 'SKIDPRO' appearing on the screen is derived from the term 'SKID PROGRAM' and represents the name given to the software.

WELCOME TO SKIDPRO

Press any key to proceed. The next screen display contains some information about SKIDPRO. This information includes what SKIDPRO is capable of doing, the names of the program developer and program director, and most importantly the copyright ownership and where it can be found. This display is shown below.
SKIDPRO

A program to determine Speed, Skidding Time, Skidding Distance and Friction Coefficient during vehicle braking. Implemented in Turbo C for MS-DOS V5.0 or newer

Developed by Abdullah S. Nuruddin as Thesis for the degree of Master of Science

Thesis/Program Director
Helmut T. Zwahlen, Ph.D

Copyright 1993, H.T. Zwahlen, Ph.D, A.S. Nuruddin, Ohio University Department of Industrial and Systems Engineering Ohio University, Athens, Ohio 45701-2979, U.S.A November, 1993

Press the <ENTER/RETURN> key to continue. The next screen displays three options, each option controlled by a function key assigned to it. By chosing and pressing the function key F1, you intend to return to the DOS prompt, chosing F2 allows you to run the program using the USA (English) units system of measurement, and by chosing F3 allows you to run the program using the Metric units measurement. To run the USA units option, two files - USA1.C and USA2.C will have to be compiled separately and then linked together at execution. Likewise, to run the Metric units option, METRIC1.C and METRIC2.C are also compiled and then linked together.
To aid in this tasks, project files are created for each units option - USA.PRJ for the USA units option and METRIC.PRJ for the Metric units option. However, the software has further simplified the task by building these two options into the UNIT.BAT file so that all that you have to do is to chose the units option you intend to work with. Within seconds after choosing the function key denoting the units you intend to work with, the program is loaded and executed and the screen display as shown below outlines the main menu options representing five different modules or sections to work with in the software.
### MAIN MENU

A. Speed  
B. Skid Distance  
C. Skid Time  
D. Coeff. of Friction  
E. Calculator  
Q. QUIT

Your selection is (A, B, C, D, E or Q) => ?

Any user input at this point has to be one of the above letters, capitalized or small. The program will not respond to any entry outside the letters listed in this menu. When option 'A' is entered, a 'SPEED' sub menu appears on the screen with a list of letters representing options to choose from.

### SPEED MENU

A. Initial Speed  
B. Final Speed  
C. Return to Main Menu  
Q. QUIT

Your selection is (A, B, C or Q) => ?
Like the main menu options, the user response is only to be one of the letters listed under the menu box. In this 'Speed' menu, you either wish to find the initial speed of a vehicle just prior to wheel lock, in which case you enter the option 'A' or that you wish to find the final speed at the end of braking, in which case you enter option 'B'. If for any reason you change your mind about finding any of the speeds and wish to return to the main menu, you enter option 'D' or option 'Q' if you wish to quit the program. The rest proceeds with an interactive question session to which you must respond to in order to get the output you are expecting from the selections and responses.

The last two menus you will encounter after the above procedures are the 'Road surface' and 'Grade of road' menus. The 'Road surface allows you to select or indicate the number of surfaces involved in a skidding scene. The software allows a maximum of three different surfaces. Below is a display of the 'Road surface' menu.

*** ROAD SURFACE ***

A. One surface
B. Two surfaces
C. Three surfaces
D. Return to Speed Menu

Your selection is (A, B, C, or D) = > ?
If you chose option A, B, or C, the 'Grade of road' menu appears, otherwise, choosing option 'D' indicates that you want to return to 'Speed' menu. The 'Grade of road' menu allows you to chose the grade of the skidding distance, i.e. whether it is level, downhill or uphill. When you enter option 'A' for level grade, the program automatically assigns a value of zero to the grade. When option 'B' is chose, you will be told that downhill grade requires a positive value and a request to enter that value. Likewise, entering option 'C' will require a negative value.

*** GRADE OF ROAD ***

A. Level  
B. Uphill  
C. Downhill  
D Return to Road surface

Your selection is (A,B, C or D) = > ?

The interactive nature of the program makes it easy to follow and use. Its user-friendliness is so designed for a wide range of user population. Its only requirement is a basic understanding vehicle dynamics including the natural laws of motion. Also the user must have a basic idea on how to get started on a computer and the ability to read and understand the information displayed on the computer screen.