SKIDPC 2000

A MOBILITY, PRODUCTION, AND RUT DEPTH SIMULATION PROGRAM

By

Ben D. Spong

In partial fulfillment of the requirements
for the degree of

Master of Forestry

Presented to:
Department of Forest Engineering
Oregon State University
Corvallis, OR 97331

October 9, 2001

ABSTRACT

SkidPC 2000, written in Visual Basic Version 6.0, is an updated version of the 1987 DOS program that calculates the mobility and productivity of ground-based logging vehicles over a ground profile. As in the original program, the user enters the vehicle and operational conditions and the program will calculate the vehicle speed, productivity, and other aspects of the vehicle, log, and soil interaction.

New features have been added to the program and there has been a complete revision of the program's user interface. Additions to the program include the ability to model six-wheeled forwarders and clambunk skidders, and the ability to simulate wheel and track rutting. Throughout the updating of the program, small errors in the original version were corrected and the user interface was restructured and streamlined to utilize the features of the Microsoft Windows environment.

SkidPC 2000 is an updated tool for operations planners, loggers, foresters, students, equipment salespeople, and others to estimate productivity, ground pressure, velocity, and wheel or track slip. This program can be used to examine relative changes in operational performance to vehicle configuration, vehicle loading, and terrain conditions.

Approved:

J. Sessions, Professor Date

TABLE OF CONTENTS

ABSTRACT ..... **II**

TABLE OF CONTENTS ..... **III**

INTRODUCTION **1**

PURPOSE 1
ORGANIZATION 2

BACKGROUND **3**

METHODS **6**

FORWARDER/CLAMBUNK VEHICLE MOBILITY 6
SOIL RUT DEPTH ESTIMATION 8
USER INTERFACE 8
STANDARDIZATION OF DRIVELINE EFFICIENCY 9
MEAN CONTACT STRESS 9

FUTURE EXTENSIONS **10**

CONCLUSIONS **13**

REFERENCES **14**

APPENDIX A: REFERENCE MANUAL **16**

INTRODUCTION **17**

PROGRAM LAYOUT **18**

PROGRAM EXTENSIONS **18**

SOIL RUT DEPTH ESTIMATION **18**
FORWARDER/CLAMBUNK VEHICLE MOBILITY **19**
USER INTERFACE **21**
STANDARDIZATION OF DRIVELINE EFFICIENCY **22**
MEAN CONTACT STRESS **22**

CONCLUSIONS **23**

REFERENCES **25**

APPENDIX **27**

LOG FORCES EQUATIONS **27**
TRACKED SKIDDER EQUATIONS **28**
RUBBER-TIRED SKIDDER EQUATIONS **34**
FORWARDER/CLAMBUNK SKIDDER EQUATIONS **39**
RUT DEPTH EQUATIONS **43**

APPENDIX B: SELECTED CODE **46**

TRACKED SKIDDER MODULE 46
RUBBER-TIRED SKIDDER MODULE 54
RUT DEPTH 66
FORM SAMPLES 70

APPRENDIX C: ONLINE HELP DOCUMENT **73**

SkidPC 2000

INTRODUCTION

PURPOSE

Ground-based logging systems are used in forest operations for extracting timber from the forest for delivery to a roadside landing. Often the ground conditions of the site can limit the continued movement of ground logging vehicles because of steep slopes or weak soils. Soil disturbance, machine damage, and even human injury can occur if a ground-based logging machine is used outside the conditions it has been designed. Using the simulation program SkidPC 2000, users can describe the conditions of the site and the machine specifications, and the program will estimate the mobility, speed, and soil disturbance.

SkidPC 2000 was developed to estimate the mobility of three types of vehicles: tracked skidders, rubber-tired skidders, and six wheeled forwarders and clambunk skidders. The program reports skidding resistances due to loading, speed, slip, and other values. These help the planner or operator understand if the appropriate machine is being used at the described location. Soil rut depth estimates are calculated using a model by Wronski and Humphreys (1994). Their model uses the cone index, vehicle loading, and the number of wheel or track passes to estimate sinkage. SkidPC 2000 results have not been field validated. All results must be carefully interpreted, especially looking at accuracy of the user-supplied data.

The program is written for the Microsoft Windows operating system and is compatible with Windows95 (or newer operating system). The results from the program can be sent to storage, a printer, or to Microsoft Excel for further calculations.

ORGANIZATION

The format for this report describes the program development process and other work completed on the project. Included with this document is a "Reference Manual" that describes the equations, references, and technical information about the program. Finally, three code modules have been included as an example of how the referenced equations were organized in the programming process. Both the reference manual and the code samples can be found in the Appendix.

BACKGROUND

SkidPC 2000 builds on a long history involving many researchers and a significant amount of work before this newest update. The concept for this program originated with a rubber-tired skidder model developed by Olsen and Gibbons in 1983. A few years later, a tracked skidder model was developed by Gleason, 1986, using the NATO mobility models. In 1987, John Balcom, a Department of Forest Engineering Research Assistant at Oregon State University, combined these two models into a single software package for the personal computer. The resulting program, SkidPC was designed for use by planners and loggers to estimate mobility and productivity of ground-based logging vehicles and user supplied terrain and load.

Since 1987, many changes have occurred in the logging industry that affects the way ground-based logging operations are planned and implemented. Changes in the timber supply have increased the importance of small diameter trees, especially in thinning operations. The small sizes of these trees make it necessary to cut more trees daily, over a larger area in order to maintain the same overall volume harvested. One method of accomplishing this is by using cut-to-length harvesters, forwarders, or feller-bunchers with clambunk skidders. Originally, SkidPC was not designed to simulate forwarders and clambunk skidders.

Minimizing the impact of these machines over the site often requires the use of designated skid trails to concentrate the machine passes to a small portion of the total harvest area. This practice minimizes impacts to the overall site, however the concentration of vehicle traffic can also increase rutting and soil compaction on the designated trails. Forest regulations, such as the Oregon Forest Practices Rule Guidance manuals, have rut depth guidelines for forest operations, which place limits on the acceptable depth of soil rutting on a skid trail. The original version of SkidPC was not designed to provide estimates of track and wheel sinkage during multiple passes under bare soil or slash over soil conditions.

SkidPC 2000 addresses many of these newer issues in ground-based logging operations, as well as new advances in computer technology and increased interest in the environmental impacts of logging operations. The program has been updated to use many of the resources of the Microsoft Windows operating system. Forwarder and clambunk options have been added to increase equipment options. Furthermore, the updated program uses a model developed by Australian researchers, Wronski and Humphrey, that estimates rut depth, using machine specifications and site conditions used in the mobility calculations. (Wronski and Humphries, 1994) The rut depth model was developed specifically for logging equipment, on forest soils and terrain. The model also adjusts the rut depth based on the amount of logging slash accumulated on the skid trail.

No validation has been done with the results of SkidPC 2000. The user must interpret all results carefully in order to avoid over or under estimating vehicle performance and production. Many of the results from the program are calculated using user-supplied data that, if not accurate, can significantly impact the final values. Additionally, the models that are used do not account for the possibility of variability that occurs in logging operations. In other words, with all the same data entered into the program, the same results will be produced and the natural variability from all the data is not recognized by the equations. The user must recognize that any results are strictly estimates and it is possible that another value around this estimate could be the actual solution.

METHODS

SkidPC 2000 has been updated to include additional functions and uses. To expand ground-based system options, a new program extension was added to model six-wheeled forwarders, clambunk skidders, and other vehicles with a front axle and a rear bogie configuration. Other additions that will be described in this section include soil rut depth estimation, the user interface, the standardization of the driveline efficiency, and the mean contact stress.

FORWARDER/CLAMBUNK VEHICLE MOBILITY

This module is based on the rubber-tired skidder module with adaptations to model the additional wheels and the different location of the load in relation to the wheels. An assumption was made in this model that the distribution of the load between the two wheels of a bogie is equal, which is the case when using a balancer or other load dividing mechanism. The model assumes the use of normal log skidder tires, not higher-pressure tires with bogie tracks. The forwarder/clambunk vehicle module is able to perform the same mobility, productivity, rut depth, and other calculations that are used in the rubber-tired skidder model. The model requires users to enter time for loading, unloading, and

Figure 1: Forwarder

Figure 2: Clambunk Skidder

delays in order to calculate the productivity of the machine. It is common for this combined loading, unloading, and delay times to be larger than the calculated moving travel time of the vehicle, so users must correctly estimate these values and interpret the resulting productivity values.

Forwarders and clambunk skidders usually load themselves as they travel along a skid trail. SkidPC 2000 will calculate the total round trip time, by adding the user supplied load, unload, and delay time to the calculated travel time based on modeled vehicle speed. The user must supply loading and unloading times. The loading time includes the time to load the logs including travel between intermediate log pickup points.

For instance, if a forwarder is working on a long skid trail, starting at the furthest point from the landing, the operator will slowly progress towards the landing while loading the vehicle with some logs, then moving a short distance and then loading additional logs. Once the forwarder’s bunks are full, the operator is assumed to move directly to the landing. In order to analyze the productivity of this system using SkidPC 2000, first an estimate on the time required to completely load the bunks of the forwarder at specific sites, including any travel time it would take to move between log concentrations. Next, the time required to fully unload the forwarder plus any delay times should be estimated. Finally, a profile should be entered into SkidPC that starts at the point along the skid trail where the vehicle became fully loaded, and continue to the landing. The distance that the forwarder traveled before being fully loaded will not be analyzed for mobility or travel time—the user estimate of this time is used here as the amount of time that the forwarder works at this loading site. Total turn time will then be calculated combining the user estimated load, unload, and delay times for the forwarder's partial loading, unloading pattern, and the calculated moving travel time between the fully loaded position and the landing. The resulting estimate of the productivity is only applicable to this single load on this skid trail. In order to properly estimate overall productivity for a sale unit, SkidPC 2000 runs for different trails that represent the spatial distribution of the trails and logs, including volume should be appropriately weighted.

SOIL RUT DEPTH ESTIMATION

SkidPC 2000 has a new module that estimates soil rut depths for each of the modeled vehicle types. The rut depth option allows the user to explore the interactions between the variables affecting soil rutting. Then they are able to make any adjustments to the input data such as soil cone index, in order to reflect the field-observed results. Using research by Wronski and Humphrey (1994) the program quantifies the relationship between slash in the skid trail, soil strength (measured by the cone index value), and wheel loading. Inputs of the cone index, tons of slash in the trail, and the maximum acceptable rut depth, allows the program to estimate the maximum number of round trip passes that are possible given the loading, vehicle, and the soil conditions. The rut depth calculations in SkidPC 2000 apply only to skidders with four wheels, forwarders and clambunk skidders with six wheels, and tracked skidders. The Wronski and Humphrey data were developed for cone index ranges between 80 and 240 psi and slash amounts in the skid trails between 0 and 18 tons/acre.

USER INTERFACE

Many changes were made to the program preferences and the user interface. SI or Standard US units can be used. Another improvement allows the program to be used on all of the Windows operating systems, which was a by-product of using Microsoft's Visual Basic programming language. Utilizing the Windows operating system, the user interface has been updated to have the look and feel of Windows application programs. This includes the use of printers, saving files, loading files, on-screen help features, and other features familiar to most users. Pop-up windows remind users to provide necessary data or fix possible problems in the user-supplied data. The initial results or the results from the rut depth analysis can either be printed to any printer or exported to Microsoft Excel with the click of a button. If the data is sent to Excel, further analysis and the compilation of multiple skid trail analyses in a single workbook is possible. Finally, all of the documentation for the program is integrated in a Help file that is accessed from a pull down window or icon.

STANDARDIZATION OF DRIVELINE EFFICIENCY

During the program revisions, input terminology was standardized and the interface was simplified so that the user is now asked to supply the efficiency of the entire driveline system of the machine. In order to maintain consistency between all modules of the program, the user supplied transmission efficiency variable has replaced the other inputs regarding the driveline system in each of the vehicle modules.

MEAN CONTACT STRESS

The original program calculated the minimum tire pressure to maintain a constant contact area. In both the rubber-tired skidder and forwarder/clambunk skidder modules, this has been replaced by an estimate of the average ground pressure of the tire on the ground or mean contact stress. The equation, from Koolen and Kuipers (1983), is based on rule of thumb constant values, tire inflation pressure, tire stiffness (ply rating), and refers to a tire on a rigid surface.

FUTURE EXTENSIONS

During the development process of SkidPC 2000, additional issues were raised for possible inclusion in future versions of the program. Six of these are described below:

1) **Braking Analysis**: Vehicle speed can be limited not by the power of the vehicle, but rather on the ability to be kept under control on downhill slopes, to stop due to unexpected conditions, or arrival at the landing. One condition that limits the maximum speed that can be sustained is the braking capacity. This speed can be calculated given the braking capacity, grade, and motion resistance. A second condition in calculating the maximum speed is the calculation of the maximum deceleration rate. The calculation of the limiting velocity of the vehicle is based on the vehicle's ability to come to a stop in a specified stopping distance given the ground conditions, slope, and vehicle loading and configuration. The user could supply the necessary information for the vehicle, estimate both limiting velocities, and then select the slowest as the overall limiting or maximum velocity for the vehicle.

2) **Maximum Load for Given Conditions**: Computer programs used for timber harvest planning, such as LoggerPC, calculate the maximum possible load given the machine and terrain conditions. This is a different approach than what is used in SkidPC 2000, where the user supplies the average load size, and the vehicle mobility is calculated for this average load. SkidPC 2000 could be modified to predict the maximum load possible to

skid, but there are a number of conditions that must considered in order to find the maximum payload on a given trail. Some of these conditions are:

- Overturning: The maximum load must keep all wheels on the ground.
- Excessive slip: The maximum load must keep the wheel or track slip under a specified maximum to avoid excessive soil disturbance, fuel use, and tire wear.
- Tire and Axle loads: The maximum load must not exceed the rated tire and axle loads.
- Stopping distance/brake: The maximum load must allow the vehicle to stop in an adequate distance for safety.
- Excessive rutting: The maximum load must not result in excessive soil rutting.

3) **Bogie track assumption**: On many forwarders and clambunk skidders, it is common to use a track around a bogie set of wheels in order to produce greater traction and lower ground pressures. SkidPC 2000 does not model these tire tracks. Further research could include this option. Some potential approaches include the tracked bogie tires in the model are:

   a) Assume tire tracks act as a simple rigid track, but then use a downgrade factor to compensate for the differences between the conceptual model and the bogie/track conditions.
   b) Assume that the track around the bogies has a known footprint and that its properties are similar to a single large wheel that has an equal footprint to the track.
   c) Assume that the track acts similarly to a short rigid track, without grousers. This would decrease the tracked bogies interaction with the soil and give results that are more than the two individual tires, but less than a rigid tracked machine with grousers.

4) **Use of Grapples**: The use of fixed grapples that are attached to the vehicle (also referred to as rigid or non-swinging) and used in the loaded condition at the approximate

height of a normal fairlead can be simulated with the existing program as an approximation. This simulation assumes that the correct center of gravity and load points for the grapple are entered into the program. This approximation becomes less appropriate if higher grapple points are used. Further research on grapple use would help expand the use of SkidPC 2000 to vehicles with the capability to position the grapple at many different heights.

5) **Rigid Log Length Skidding Model**: SkidPC 2000 uses a whole tree length log model for calculating log skidding resistances. Including a rigid log model option would add an alternative relationship for shorter rigid log lengths. This model would include a calculation for the center of gravity of the log.

6) **Improved Mobility Equations**: SkidPC 2000 uses the original equations from Ashmore, et al (1985) to calculate the thrust of rubber tire skidders and six wheeled vehicles. An update of the mobility equations to include the equations published by Brixius (1988) and the revised Brixius and Ashmore equations (Rawlins, et al (1996)) could provide the user more flexibility in choosing the appropriate equations. The revised equations differentiate between cohesive and non-cohesive soils.

CONCLUSIONS

The objective of this project was to update and improve SkidPC, a ground vehicle mobility computer program. SkidPC 2000 is the result of this update and demonstrates many changes to the original version. SkidPC 2000's updated features are as follows:

1. The new program is Microsoft Windows compatible and can be used on most computers. A simple, intuitive design makes the program easy to use and easy to troubleshoot.

2. Additional vehicles are modeled in the new program. With the addition of six-wheeled forwarders and clambunk skidders, along with conventional tracked and rubber-tired skidders, more ground based machines are included in the program.

3. The vehicle ground pressure calculations have been added to estimate the average ground pressure under the tires on rubber-tired skidders, forwarders, and clambunk skidders.

4. Soil rut depth and slash effects on rut depth are estimated for all machine types. The user is able to estimate the number of vehicle passes that are possible before a given rut depth is created.

5. The program now supports additional output options including printing, export to Microsoft Excel, and saving to disk.

REFERENCES

Ashmore, C., E.C. Burt, and J.L. Turner, 1985. Predicting Tractive Performance of Log-Skidder Tires. American Society of Agricultural Engineers Paper No. 85-1597.

Brixius, W.W. 1987. Traction Prediction Equations for Bias Ply Tires. American Society of Agricultural Engineers Paper No. 87-1622.

Gleason, K.H., 1985. A Mobility Model for Tracked Vehicles. Master of Forestry paper; Oregon State University. 150p.

Herrick, D.E., 1955. Tractive Effort Required To Skid Hardwood Logs. Forest Products Journal 5(4) : 250 – 255.

Koger, J.L., E.C. Burt, A.C. Bailey, 1985. Load Deflection Relationships for Three Log Skidder Tires. Research Note U.S. Dept. of Agric. Forest Service, Southern Forest Research Station; . p. 4.

Koolen, A.J., H. Kuipers, 1983. Agricultural Soil Mechanics. Springer-Verlag, Berlin, Heidelberg. P. 136.

Krick, G., 1969. Druck-und Schubverteilung unter Rädern und Reifen auf nachgiebigem Boden unter Berücksichtigung der Reifendeformation. Proc 3rd Int Conf ISTVS, Essen, Vol 2, pp 50–75.

Olsen, E.D., D.J. Gibbons, 1983. Predicting Skidder Productivity: A Mobility Model. Research Bulletin 43, Forest Research Lab, Oregon State University, 19p.

Oregon Department of Forestry, 2001. ODF Interpretive Guidance–Forest Practices Act, Website:http://www.odf.state.or.us/FP/RefLibrary/Rule&LawGuidance/Administrative%20Rules%20and%20Guidance/Division%20630%20%20Harvesting.doc

Perumpral, J.V. 1977. Skidding forces of tree length logs predicted by a mathematical model. Transactions of the American Society of Agricultural Engineers 20(6):1008-1012.

Rawlins, C., E.C. Burt, and C.E. Johnson, 1989. Traction Prediction Equation Coefficients for Forestry Tires. American Society of Agricultural Engineers Paper No. 89-1556.

Sessions, J., 1999 Logging Mechanics Notes (FE 571). Oregon State University, p. 6.

Sessions, J. and W. Chung. 2001. Skidding forces for Grapple Skidders. Submitted to Forest Products Journal. 6 p. 2 fig.

Wismer, R.D., and H.J. Luth. 1974. Off-road traction prediction for wheeled vehicles. Transactions of the American Society of Agricultural Engineers 17(4):8-10, 14.

Wronski, E.B., N. Humphreys, 1994. A Method for Evaluating the Cumulative Impact of Ground-Based Logging Systems on Soils. Journal of Forest Engineering, January 1994, 9 -20.

APPENDIX A: REFERENCE MANUAL

SKIDPC 2000

REFERENCE MANUAL

A MOBILITY, PRODUCTION, AND RUT DEPTH SIMULATION PROGRAM

By

Ben D. Spong

Presented to:
Department of Forest Engineering
Oregon State University
Corvallis, OR 97331

October 9, 2001

SkidPC 2000

REFERENCE MANUAL

INTRODUCTION

SkidPC is a mobility and productivity computer simulation application for ground-based log skidding machines. The concept for this program originated with a rubber-tired skidder model developed by Olsen and Gibbons in 1983. This work was then combined with a model for tracked skidders developed by Gleason in 1986. In 1997, John Balcom, with the Department of Forest Engineering at Oregon State University, put these two models together into a single software package for the personal computer. The resulting program was SkidPC and was designed for use by planners and loggers to estimate mobility and productivity of ground-based logging vehicles and user supplied terrain and log load.

Since 1987, there have been significant changes in logging systems, computer technology, and interest in the environmental impacts of logging operations. An update to SkidPC has been completed that helps address many of these issues, including better use of the computer technology, modeling of additional machines, and modeling machine caused soil rutting. The new program is called SkidPC 2000. SkidPC 2000 has not been field validated. All results must be carefully interpreted, especially looking at accuracy of the user-supplied data.

PROGRAM LAYOUT

The SkidPC 2000 program is set up in three different modules: tracked skidder, rubber-tired skidder, and clambunk/forwarder modules. In each of these modules, the steps are similar--the user selects the module and then is asked to select a predefined vehicle or enter the specifications for a new vehicle. Next, the user is asked to enter the operational conditions and ground profile of the site. Once the user supplied data has been entered, the program calculates and reports an estimate of the mobility of the machine over the desired terrain, and gives an estimate of the productivity based on the machine's mobility and the operational efficiency. Results are displayed to the screen and can be sent to the printer. The user can exit the program here, or continue with an option to perform a soil rut depth analysis using the same vehicle and ground profile.

PROGRAM EXTENSIONS

The first version of SkidPC offered users a tool for estimating mobility and productivity. However, over time, it became clear that additional extensions to the program and improvements that would correct small errors in the original version could expand the usefulness and application of the program.

SOIL RUT DEPTH ESTIMATION

The ability to model some of the environmental effects of log skidding allows the planner or equipment operator to estimate the number of passes over a skid trail section before meeting a user supplied maximum rut depth. In some timber sale contracts, a maximum allowable rut depth is specified, where if passed the operation is stopped until soil conditions improve or other mitigation occurs. The rut depth option in SkidPC 2000 allows the user to explore the interactions between variables affecting the soil rut depth. Additionally, the user is able to make adjustments to the input data such as soil cone index, in order to reflect the field observed results. SkidPC 2000 uses a relationship between soil strength (measured by a cone index value) and wheel loading that was developed by Wronski and Humphreys (1994). They also observed that there was a decrease in soil rut depth when slash was added to the skid trail. To quantify this relationship in SkidPC 2000, the amount of slash (tons/acre) in the skid trail is converted into a "slash factor" that decreases the effective pressure on the soil. The user is able to indicate the cone index, the amount of slash, and the maximum acceptable rut depth, then the program will calculate the maximum number of round trip passes that are possible given the loading and the soil conditions. The rut depth module is available in all three vehicle modules and is run after the original vehicle analysis has been completed.

FORWARDER/CLAMBUNK VEHICLE MOBILITY

An option to model six-wheeled forwarders, clambunk skidders, and other vehicles with a front axle and a rear bogie configuration has been added to the program. This module is based on the rubber-tired skidder module with adaptations to model the additional wheels and the different location of the load in relation to the wheels. An assumption was made in this model that the distribution of the load between the two tires of a bogie pair is equal. This distribution is often done with a balancer or other load dividing mechanism. The model assumes the use of normal skidder type tires, not bogie tracks. The user can perform the same mobility, productivity, rut depth, and other calculations with the forwarder/clambunk module as in the rubber-tired skidder model.

The model requires users to enter time for loading, unloading, and delays in order to calculate the productivity of the machine. It is common for this combined loading, unloading, and delay times to be larger than the calculated moving travel time of the vehicle, so users must correctly estimate these values and interpret the resulting productivity values.

Forwarders and clambunk skidder usually load themselves as they travel along a skid trail. SkidPC 2000 will calculate the total round trip time, by adding the user supplied load, unload, and delay time to the calculated travel time based on modeled vehicle speed. Therefore the user must supply loading and unloading times, recognizing the unique pattern of driving to loading point, partially loading the log bunks, and then driving to next loading point until a load is achieved.

For instance, if a forwarder is working on a long skid trail, starting at the furthest point from the landing, the operator will slowly progress towards the landing while loading the vehicle with some logs, then moving a short distance and then loading some more logs. Once the forwarder's bunks are full, the operator is assumed to move directly to the landing. In order to analyze the productivity of this system using SkidPC 2000, first an estimate on how long it will take to completely load the bunks of the forwarder at specific sites, including any travel time it would take to move between log concentrations. Next, the time required to fully unload the forwarder plus any delay times should be estimated. Finally, a profile should be entered into SkidPC that starts at the point along the skid trail where the vehicle became fully loaded, and continue to the landing. The distance that the forwarder traveled before being fully loaded will not be analyzed for mobility or travel time—the user estimate of this time is used here as the amount of time that the forwarder works at this loading site. Total turn time will then be calculated combining the user estimated load, unload, and delay times for the forwarder's partial loading, unloading pattern, and the calculated moving travel time between the fully loaded position and the landing. The resulting estimate of the productivity is only applicable to this single load on this skid trail. In order to properly estimate overall productivity for a sale unit, SkidPC 2000 runs for different trails that represent the spatial distribution of the trails and logs, including volume should be appropriately weighted.

USER INTERFACE

Many changes were made to the program preferences and interface. The user now is able to choose either SI or Standard US units. On the first screen of the program, the user chooses "Standard Units (FPS)" or "SI Units" by clicking the appropriate button. The program will then process all the input and output in the units selected at the beginning of that run. The program was written in Microsoft's Visual Basic Version 6.0 programming language, for use on all of the Windows operating systems. Utilizing the Windows operating system, the user interface has been updated to have the look and feel of other Windows Applications. This includes the use of printers, saving files, loading files, help features, and many other features that users are already familiar with. Pop-up windows remind users to provide necessary data or fix possible problems in the user-supplied data. The results from the analyses can be printed to any printer or exported directly to Microsoft Excel with the click of a button for further analysis and compilation of many runs in a single file. Finally, all documentation for the program is integrated in a Help file, retrievable by clicking an icon or pull down menu.

STANDARDIZATION OF DRIVELINE EFFICIENCY

During the revisions, terminology and input conventions were improved. One update was the standardization of the driveline efficiency of each of the vehicle modules, but specifically in the tracked skidder module. To recognize the lower efficiency of fluid transmissions, the user is now asked to supply the efficiency of the entire transmission system for the machine. This combines all the different options and accessories that might have an impact on the overall power that is ultimately transferred to the drive wheels. In order to maintain consistency between all modules of the program, the user supplied transmission efficiency variable has replaced all other inputs regarding the transmissions – torque converters, powershift, and others for all modules including tracked skidder, rubber-tired skidder, and forwarder/clambunk skidder modules.

MEAN CONTACT STRESS

Another revision concerns the calculated minimum tire inflation that was reported in the analysis of the rubber-tired skidders. This was the minimum tire pressure to maintain a constant contact area. This has been replaced by an estimate of the mean contact stress. It provides the user with an estimate of the ground pressure of the tire on the ground, as a function of inflation pressure and tire stiffness (Koolen and Kuipers, 1983). A preliminary attempt at finding a solution started by calculating the deflection of the tire under the load (Koger, et al, 1985), then calculating the deflected tire contact area (Krick, 1969), and finally dividing by the normal force to give a mean contact stress in pounds per square inch. The results provided by these calculations consisted of values for the mean contact stress being less than the tire inflation pressure. In almost all rubber tire skidder and forwarder configurations and situations, we would expect the mean contact stress would be greater than the tire inflation pressure. As an alternative approach to provide an estimate of the mean contact stress, a rule of thumb equation and constants are used to calculate out the mean contact stress on a rigid surface based on the ply rating of the tire and the tire inflation pressure (Koolen and Kuipers, 1983). These calculations are done in both the rubber-tired skidder and forwarder/clambunk skidder modules.

CONCLUSIONS

SkidPC 2000 is a complete update of the original SkidPC program, including many enhancements that make the program more robust and easier to use. The best way to get a feel for the program is to try it. Once the program has been loaded, the user can experiment with sample data that is included in the program and use the help files to guide them through the program. Supplemental information, including a description of the basic equations used in the program can be found in the appendix to this Reference Manual. A sample of the code from each of the vehicle modules is also included in the Master of Forestry paper submitted to the Department of Forest Engineering at Oregon State University.

SkidPC 2000 provides a tool that can be used by a variety of people who would like to understand the inter-relationships between vehicle parameters, ground, and log load, and how they affect mobility, productivity, and environmental effects of log skidding.

REFERENCES

Ashmore, C., E.C. Burt, and J.L. Turner, 1985. Predicting Tractive Performance of Log-Skidder Tires. American Society of Agricultural Engineers Paper No. 85-1597.

Brixius, W.W. 1987. Traction Prediction Equations for Bias Ply Tires. American Society of Agricultural Engineers Paper No. 87-1622.

Gleason, K.H., 1985. A Mobility Model for Tracked Vehicles. Master of Forestry paper; Oregon State University. 150p.

Herrick, D.E., 1955. Tractive Effort Required To Skid Hardwood Logs. Forest Products Journal 5(4) : 250 – 255.

Koger, J.L., E.C. Burt, A.C. Bailey, 1985. Load Deflection Relationships for Three Log Skidder Tires. Research Note U.S. Dept. of Agric. Forest Service, Southern Forest Research Station; . p. 4.

Koolen, A.J., H. Kuipers, 1983. Agricultural Soil Mechanics. Springer-Verlag, Berlin, Heidelberg. P. 136.

Krick, G., 1969. Druck-und Schubverteilung unter Rädern und Reifen auf nachgiebigem Boden unter Berücksichtigung der Reifendeformation. Proc 3rd Int Conf ISTVS, Essen, Vol 2, pp 50–75.

Olsen, E.D., D.J. Gibbons, 1983. Predicting Skidder Productivity: A Mobility Model. Research Bulletin 43, Forest Research Lab, Oregon State University, 19p.

Oregon Department of Forestry, 2001. ODF Interpretive Guidance–Forest Practices Act, Website:http://www.odf.state.or.us/FP/RefLibrary/Rule&LawGuidance/Administrative%20Rules%20and%20Guidance/Division%20630%20%20Harvesting.doc

Perumpral, J.V. 1977. Skidding forces of tree length logs predicted by a mathematical model. Transactions of the American Society of Agricultural Engineers 20(6):1008-1012.

Rawlins, C., E.C. Burt, and C.E. Johnson, 1989. Traction Prediction Equation Coefficients for Forestry Tires. American Society of Agricultural Engineers Paper No. 89-1556.

Sessions, J., 1999 Logging Mechanics Notes (FE 571). Oregon State University, p. 6.

Sessions, J. and W. Chung. 2001. Skidding forces for Grapple Skidders. Submitted to Forest Products Journal. 6 p. 2 fig.

Wismer, R.D., and H.J. Luth. 1974. Off-road traction prediction for wheeled vehicles. Transactions of the American Society of Agricultural Engineers 17(4):8-10, 14.

Wronski, E.B., N. Humphreys, 1994. A Method for Evaluating the Cumulative Impact of Ground-Based Logging Systems on Soils. Journal of Forest Engineering, January 1994, 9 -20.

APPENDIX

LOG FORCES EQUATIONS

The first step of the analysis is to calculate the log forces using the equations from Herrick (1955) and Perumpral (1977) and extended by Olsen and Gibbons (1983).

Normal Force of Tree Length Log on Soil

$$q_{Arch} = \frac{W1\left[(Cg - Lb)\sqrt{1-(H/x)^2} * \cos(T) + H(Y/x - 1)\sin(T)\right]}{x\sqrt{1-(H/x)^2} + \mu_s H + (Lc/2)}$$

$$q_{NoArch} = C * TurnWt * Cos(T)$$

q	=	Normal force of Tree Length on Soil (with arch or no arch), lb.
Cg	=	Distance from log butt to center of gravity, ft.
Lb	=	Distance from butt to choker hook point, ft.
H	=	Distance from ground to hook point, ft.
x	=	L1 - Lb – Lc, ft.
L1	=	Total log length, ft.
Lb	=	Distance from butt to choker hook point, ft.
Lc	=	Average length of contact of log and ground, ft.
μ_s	=	Coefficient of skidding resistance
T	=	Ground slope, degrees
C	=	Weighting factor from empirical research (C=0.8)

Reference: Olsen and Gibbons (1983)

Normal Force of Logs

$$P1_{NoArch} = TurnWt * Cos(T) - q$$

$$P1_{Arch} = 0.2 * TurnWt * Cos(T)$$

P1	=	Normal force of logs (with and without arch), lb.
TurnWt	=	Weight of turn of logs, lb.
T	=	Ground slope, degrees
q	=	Normal force of tree length log on soil, lb.

Herrick (1955)

Tangential Force of Logs

$$P2_{NoArch} = 0.8 * TurnWt * Cos(T) + TurnWt * Sin(T)$$
$$P2_{Arch} = TurnWt * Sin(T) + u * q$$

P2	=	Tangential force of logs (with and without arch), lb.
TurnWt	=	Weight of turn of logs, lb.
T	=	Ground slope, randians
u	=	Coefficient of skidding resistance
q	=	Normal force of tree length log on soil, lb.

Reference: Herrick (1955)

Maximum Winchline tension

$$\mathrm{MAXTENS} = \sqrt{[\mathrm{C} * (\mathrm{P2} - \mathrm{TurnWt} * \mathrm{Sin(T)}) + TurnWt * Sin(T)]^2 + P1^2}$$

MAXTENS	=	Winchline tension during breakout (maximum), lb.
P1	=	Component of winchline tension perpendicular to the ground, lb.
P2	=	Component of winchline tension parallel to the ground, lb.
TurnWt	=	Weight of Turn of logs, lb.
T	=	Ground Slope, degrees
C	=	Constat that represents the breakout force of 1.1 times the skidding resistance, lb.

Reference: Olsen and Gibbons (1983)

TRACKED SKIDDER EQUATIONS

SkidPC 2000 takes the user supplied machine specifications and operating conditions and performs calculations estimating the mobility and production for the unloaded machine and then carries out the analysis for the loaded machine. An annotated description of the loaded machine equation set is presented here. After calculating the log forces, SkidPC 2000 calculates the Normal force and resultant location for the vehicle and load.

Pressure Distribution of Tracked Skidder

$$Xr = \frac{CrawlerW * Xc * Cos(T) + CrawlerW * Yc * Sin(T) + P1 * Xp + P2 * Yp}{GCW}$$

Xr	=	Resultant Location from center of front wheel, decimal
Crawler W	=	Weight of the tracked skidder, lb.
Xc	=	Center of Gravity-Horizontal distance from front of track, in.
Yc	=	Center of Gravity-Vertical height above ground, in.
T	=	Ground Slope, degrees
Xp	=	Hook Point-Horizontal distance from front of track, in.
Yp	=	Hook Point-Vertical height above ground, in.
P1	=	Normal Force of Logs, lb.
P2	=	Tangential Force of Logs, lb.
GCW	=	Gross Combined Weight, lb.

Reference: Gleason (1986)

The pressure on the leading and trailing edge of the tracks is then calculated based on where the resultant location of the pressure is located. Using these values, the contact length of a track can be calculated. The track is now divided up into segments that are delimited by the rollers or track wheels. The discrete areas are then calculated for each segment. Once each segment and its associated area is found, the reaction points at each end of the tracks are calculated using the secant method, an iterative loop, looking particularly at the sum of the static forces perpendicular to the ground and the sum of the moments about the front wheel. Finally, the track ground pressure is calculated and the results are reported (Gleason, 1987).

Next, SkidPC 2000 uses the NATO model algorithms (Gleason, 1987) to calculate the vehicle's velocity. First, looking at fine-grained soil types we can calculate the mobility index value.

Mobility index

$$XMI = (CPAVG * WF / TF / GF + WLORF - CLF) * EF * TFX$$

XMI	=	Mobility Index
CPAVG	=	Average pressure on tracks, psi
WF	=	Weight Factor – based on GCW
TF	=	Track Factor – function of track width

GF	=	Grouser Factor – function of grouser height
WLORF	=	Bogie Load Range Factor
CLF	=	Clearance Factor – function of minimum clearance
EF	=	Engine Factor – function of flywheel hp
TFX	=	Transmission Factor – based on powershift

Reference: Gleason (1986)

The mobility index and the soil cone index values are then used to calculate the drawbar pull coefficient, as limited by the soil strength. The motion resistance coefficient can also be calculated – as a function of the rating cone index. These values are now used to calculate the slip curve correction factor, or the difference between the drawbar pull coefficient at 20% slip at maximum soil strength minus the drawbar pull coefficient at 20% slip at the rating cone index of the skid trail segment being analyzed. At this point the soil limited gross tractive effort at 100% slip can also be calculated.

If the skid trail segment has a coarse-grained soil, the program would check for a flexible tracked machine. Depending on if the track was flexible or not, the pressure distribution factor for coarse-grained soils is calculated changing the constant based on track flexibility.

<u>Pressure Distribution Factor</u>

$$PIT = FtrackCon * G * \left(\frac{(TrackWd * TrackLn)^{1.5}}{GCW/2} \right)$$

PIT	=	Pressure Distribution Factor
FtrackCon	=	Constant – if flex track = 0.63 if rigid = 1.0
G	=	Coarse Grained soil penetration resisting gradient, psi
TrackWd	=	Track width, in.
TrackLn	=	Track length, in.
GCW/2	=	Gross combined weight for one track, lb.

Reference: Gleason (1986)

A drawbar pull coefficient, as limited by the soil strength, which is a function of the pressure distribution factor is calculated next. As in the fine-grained soil equations, a motion resistant coefficient, slip curve correction factor, and the soil limited gross tractive effort at 100% slip are calculated.

With all of the soil specific values calculated, the resistances must now be summed in order to calculate the tracked skidder's theoretical velocity.

Resistances

$$GR = CrawlerWt * Sin(T)$$

GR	=	Grade resistance, lb.
CrawlerWt	=	Tracked skidder weight, lb.
T	=	Ground slope, degrees

Reference: Gleason (1986)

$$RS = \frac{GCW}{2000} * C$$

RS	=	Resistance due to the internal working of the track and suspension system, lb.
GCW/2000	=	Gross combined weight converted from lbs to tons
C	=	Constant from empirical research (C=30+0.9*4)

Reference: Gleason (1986)

$$CR = (RTOWPB * GCW - RS) * CRINC$$

CR	=	Compaction resistance, lb.
GCW	=	Gross combined weight, lb.
RTOWPB	=	Motion Resistance coefficient, decimal
RS	=	Resistance due to the internal working of the track and suspension system, lb.
CRINC	=	Factor used to increase the compaction resistance from NATO, decimal

Reference: Gleason (1986)

$$RESIST = CR + FGRADE + P2$$

CR	=	Compaction resistance, lb.
FGRADE	=	Grade resistance, lb.
P2	=	Component of winchline tension parallel to the ground, lb.

Reference: Gleason (1986)

Theoretical Velocity

$$VT = \frac{V(NGEAR) + (V(NGEAR+1) - V(NGEAR))}{DP(NGEAR) - DP(NGEAR+1)} * DP(NGEAR) - RESIST$$

VT	=	Theoretical velocity, mph
NGEAR	=	Transmission gear number (1, 2, 3, 4, 5, or 6th)
V(NGEAR)	=	Velocity in transmission gear NGEAR, mph
DP(NGEAR)	=	Drawbar pull in transmission gear NGEAR, lb.
RESIST	=	Sum of compaction resistance, grade resistance, and log payload resistance, lb.

Reference: Gleason (1986)

The program then tests to see if there is enough soil tractive effort to overcome these resistances or the initial breakout forces required. The theoretical velocity is the speed the tracked skidder could go assuming no slip. If we adjust for slip in the equation, we can calculate the actual velocity. Again there are separate equations for fine and coarse-grained soils.

Slip for fine grained soils

$$YX = \frac{CR + P2 + FGRADE + RS}{GCW - CF}$$

YX	=	Corrected drawbar pull coefficient, lb.
CR	=	Compaction resistance, lb.
P2	=	Component of winchline tension parallel to the ground, lb.
FGRADE	=	Grade resistance, lb.
RS	=	Resistance due to the internal working of the track and suspension system, lb.
GCW	=	Gross combined weight, lb.
CF	=	Slip curve correction Factor, decimal

Reference: Gleason (1986)

$$SLIPXL4 = \frac{0.0257 * YX - 0.0161 + 0.01519}{(0.8353 - YX)}$$

SLIPXL4	=	Slip of the track on the ground when average pressure on tracks is less than 4.0 psi, decimal
YX	=	Corrected drawbar pull coefficient, decimal
Constant Values	=	Constants from empirical research

Reference: Gleason (1986)

$$SLIPXG4 = \frac{0.0733 * YX - 0.0063 + 0.00734}{(0.7177 - YX)}$$

SLIPXG4	=	Slip of the track on the ground when average pressure on tracks is greater than or equal to 4.0 psi, decimal
YX	=	Corrected drawbar pull coefficient, decimal
Constant Values	=	Constants from empirical research

Reference: Gleason (1986)

Slip for Coarse Grained Soils

$$SLIPX\_R = \frac{-0.0083 + 0.005312}{(0.573 - YX)}$$

SLIPX_R	=	Slip of the track on the ground with RIGID track, decimal
YX	=	Corrected drawbar pull coefficient, decimal
Constant Values	=	Constants from empirical research

Reference: Gleason (1986)

$$SLIPX\_F = YY + \sqrt{YY^2 + 0.09000001 * YX + 0.0089999999}$$

SLIPX_F	=	Slip of the track on the ground with FLEXIBLE track, decimal
YY	=	Corrected drawbar pull coefficient for coarse grained soils (YY = 1.704 * YX - 0.72), decimal
YX	=	Corrected drawbar pull coefficient, decimal
Constant Values	=	Constants from empirical research

Reference: Gleason (1986)

Actual Velocity of Tracked skidder

$$VA = VT * (1 - SLIPX)$$

VA	=	Actual velocity, mph
SLIPX	=	Slip of the track on the ground, decimal
VT	=	Theoretical velocity, mph

Reference: Gleason (1986)

RUBBER-TIRED SKIDDER EQUATIONS

Using the user supplied machine specifications and operating conditions, the first step of the analysis is to calculate the log forces using the log force equations. SkidPC 2000 then calculates the normal force for the vehicle and load

Normal Weight on Front Tires loaded

$$N1 = \frac{\left(SkidderW * \left(L - Xc\right) * Cos(T) - SkidderW * YC * Sin(T) - P1 * Xp - P2 * Yp\right)}{L} + Ballast * 2$$

L	=	Length of skidder wheel base, in.
SkidderW	=	Weight of skidder, lb.
Xc	=	Horiz. length btwn front axle and center of gravity, in.
Xp	=	Horiz. length btwn rear axle and lead point on arch, in.
Yp	=	Height of lead point on arch, in.
Yc	=	Height of center of gravity, in.
T	=	Ground slope, degrees
Ballast*2	=	Weight of ballast for two wheels, lb.

Reference: Olsen and Gibbons (1983)

Normal Weight on Rear Tires loaded

$$N2 = SkidderW * Cos(T) + P1 - N1$$

SkidderW	=	Weight of skidder, lb.
P1	=	Normal force of logs, lb.
N1	=	Normal weight from logs on front tires loaded, lb.
T	=	Ground slope, degrees

Reference: Olsen and Gibbons (1983)

Next, the gross thrust from both the front and rear wheels are calculated starting with a slip of 9%. The breakout force required is assumed to be 1.1 times the skidding resistance. Slip is increased until the breakout force is overcome.

Gross Thrust (By Axle)

$$P5 = .47\left[\left(1 - e^{-0.2*C2*S}\right) + 0.28\left(\frac{N1*0.5}{WR1}\right)\right]$$

$$P6 = .47\left[\left(1 - e^{-0.2*C3*S}\right) + 0.28\left(\frac{N2*0.5}{WR2}\right)\right]$$

P5	=	Gross thrust from two front tires (loaded), lb.
P6	=	Gross thrust from two rear tires (loaded) , lb.
C2	=	Freitag numeric front tire
C3	=	Freitag numeric rear tire
e	=	Base of natural logarithm
S	=	Wheel slip, decimal
N1	=	Normal force on front two tires, lb.
N2	=	Normal force on rear two tires, lb.
WR1	=	Rated tire loading capacity front, lb.
WR2	=	Rated tire loading capacity rear, lb.
Constant Values	=	Constants from empirical research

Reference: Ashmore, et al (1985)

$$PP = N1*(P5 - R5) + N2*(P6 - R6)$$

PP	=	Total tractive pull parallel to ground, lb.
N1	=	Normal force on front two tires (loaded), lb.
P5	=	Gross thrust from front two tires (loaded), lb.
R5	=	Rolling resistance encountered by front two tires (loaded), lb.
N2	=	Normal force on rear two tires (loaded), lb.
P6	=	Gross thrust from rear two tires (loaded), lb.
R6	=	Rolling resistance encountered by rear two tires (loaded), lb.

Reference: Olsen and Gibbons (1983)

$$F = PP - 1.10 * P2 - SkidderW * Sin(T)$$

F	=	Breakout force , lb.
PP	=	Total tractive pull parallel to ground, lb.
P2	=	Tangential force of logs, lb.
SkidderW	=	Skidder weight, lb.
T	=	Ground slope, degrees
Constant Values	=	Constat that represents the breakout force of 1.1 times the skidding resistance, decimal

Reference: Olsen and Gibbons (1983)

Once the logs are broken out, the vehicle velocity is calculated.

Velocity

$$V2 = [workpace]/[100*(1-S)*(h1*33*1000)]/[(N1*R5+N2*R6+P2+SkidderW*\sin(T)]$$

V2	=	Loaded velocity, ft/min
Workpace	=	Work pace adjustment factor, decimal
S	=	Wheel slip, decimal
h1	=	Horsepower available at axle, hp
N1	=	Normal weight on front two tires, lb.
R5	=	Rolling resistance encountered by front two tires, lb.
N2	=	Normal weight on rear two tires, lb.
R6	=	Rolling resistance encountered by rear two tires, lb.
P2	=	Tangential force of logs, lb.
SkidderW	=	Skidder weight, lb.
T	=	Ground Slope, degrees

Reference: Olsen and Gibbons (1983)

Note that the model is based on "robotics." It assumes that the machine is always being run at its optimal level (i.e. the machine is always being run in the best gear, the engine is properly tuned for the conditions, etc.). The human element and mechanical deficiencies are usually ignored in a model such as this. The work pace adjustment factor gives the user the option of bringing the human element or mechanical deficiencies back into the picture. The default valve is 100% (this assumes optimum use of the machine). If you want to reduce this value, simply input the appropriate percentage of optimum.

Mean contact stress is then calculated to estimate the average ground pressure impact of the vehicle using rule of thumb equations and constants from Koolen and Kuipers (1983).

Mean Contact Stress

$$P_m = k * p_i$$

Pm	=	Mean Contact Stress, psi
k	=	Constant from imperical research
pi	=	Tire inflation pressure, psi

Reference: Koolen and Kuipers (1983)

The *k* constant has one of four different values based on ply rating of the tire. SkidPC 2000 selects the constant value from Koolen and Kuipers that closest matches the user input ply rating. The different values that are possible are as follows:

Ply Rating	Constant
< 6	1.10
>6 and <10	1.15
>10 and <16	1.20
>16	1.25

Reference: Koolen and Kuipers (1983)

After all the calculations for the loaded condition are solved, then the program will analyze the unloaded return trip for this profile segment. The program loops through this routine until all segments are covered. Finally, travel time and productivity equations are then calculated and the results are reported.

$$t2 = \left(\frac{A2}{V2}\right) + \left(\frac{A2}{V3}\right) + T3$$

t2	=	Cumulative travel time, min.
A2	=	Slope distance, ft.
V2	=	Velocity loaded, ft./min.
V3	=	Velocity unloaded, ft./min.
T3	=	Time required to hook, unhook, and deck logs per turn, min.

Reference: Olsen and Gibbons (1983)

$$productivity = \frac{LBSPERHOUR}{LBSMBF}$$

Productivity	=	Productivity with delays,
LBSPERHOUR	=	Pounds of logs yarded per hour, lbs/hr
LBSMBF	=	Number of pounds in one thousand board feet, lbs/mbf

$$UnitCost = \frac{F2}{LBSPERHOUR * LBSMBF}$$

UnitCost	=	Cost per unit yarded, \$/mbf
F2	=	Average operating cost, \$/hr
LBSPERHOUR	=	Pounds of logs yarded per hour, lbs/hr.
LBSMBF	=	Number of pounds in one thousand board feet, lbs/mbf

It is possible to use SkidPC 2000 to estimate productivity of different equipment, operating conditions, and terrain profiles. However, it is important to understand how productivity is calculated in SkidPC 2000 and how to use the computer output to portray the operation. In the program, the calculated productivity rates are only for a single skid trail, where the vehicle loads at the end of the trail, travels to the landing (the other end of the trail), unloads all the logs, and then returns unloaded to end of the trail. The productivity rate also includes user supplied loading, unloading, and delay times. The reported productivity rates assume that these exact skid trail and operating conditions are repeated multiple times in order to estimate hourly production rates.

Many users inappropriately attempt to use the productivity results from a single skid trail and apply the same rates for the entire sale area. Many problems arise when the value from one trail, even if it is a typical or average trail, are applied to the whole unit. For instance, any difference in the distribution of logs along the skid trail or across the entire unit would change the productivity. Branching skid trail patterns and other more complex distribution of skid trails are also not represented in the productivity calculations. In order to properly estimate overall productivity for a sale unit, SkidPC 2000 runs for different trails that represent the spatial distribution of the trails and logs, including volume should be appropriately weighted.

Likewise, it is inappropriate to analyze the steepest single skid trail on the harvest unit as the one that would be limiting and then use the reported productivity rates applied to the entire unit. In this case, a vehicle much too large or expensive might be used for a unit, even though the majority of the sale could be logged using other equipment. The user would need to make multiple SkidPC 2000 runs and make decisions on vehicle selection for each portion of the sale area, the available equipment, and other important factors, e.g. timber distribution, delay times, etc.

FORWARDER/CLAMBUNK SKIDDER EQUATIONS

Using data supplied by the user on the machine specifications and the operational conditions, SkidPC 2000 uses the user supplied machine specifications to check if the logs will be fully suspended. If the logs will be carried with no ground/log interaction during the skid, as occurs when a forwarder is used, all log drag calculations are skipped. If the logs will be skidded with a clambunk skidder, where the end of the log is dragged behind the vehicle, SkidPC 2000 calculates the log forces using the log force equations previously described. SkidPC 2000 then calculates the normal force for the vehicle and load

Normal Weight on Front Tires loaded

$$N1 = \frac{(ForwarderW * (L - Xc) * Cos(T) - ForwarderW * YC * Sin(T) - P1 * Xp - P2 * Yp)}{L} + Ballast * 2$$

L	=	Length of forwarder wheel base, in.
ForwarderW	=	Weight of forwarder, lb.
Xc	=	Horiz. length btwn front axle and center of gravity, in.
Xp	=	Horiz. length btwn rear axle and lead point on arch, in.
Yp	=	Height of lead point on arch, in.
T	=	Ground Slope, degrees
Ballast*2	=	Weight of Ballast in front two tires, lb.

Reference: Olsen and Gibbons (1983)

Normal Weight on Rear Tires loaded

$$N2 = ForwarderW * Cos(T) + P1 - N1$$

ForwarderW	=	Weight of forwarder, lb
P1	=	Normal force of logs, lb
N1	=	Normal weight on front tires (loaded), lb
T	=	Ground Slope, degrees

Reference: Olsen and Gibbons (1983)

Next, the gross thrust from both the front and rear wheels are calculated using equations derived by Ashmore et al (1985). These calculations started with a minimum assumed slip of 9%, which was increased until the breakout force is overcome. The breakout force required is calculated to be 1.1 times the skidding resistance.

Gross Thrust

$$P5 = .47\left[\left(1 - e^{-0.2\times C2\times S}\right) + 0.28\left(\frac{N1 * 0.5}{WR1}\right)\right]$$

$$P6 = .47\left[\left(1 - e^{-0.2\times C3\times S}\right) + 0.28\left(\frac{N2 * 0.5}{WR2 * 2}\right)\right]$$

P5	=	Gross thrust from front two tires (loaded), lb.
P6	=	Gross thrust from rear four tires (loaded), lb.
C2	=	Freitag numeric front tire
C3	=	Freitag numeric rear tires (one bogie)
e	=	Base of natural logarithm
S	=	Wheel slip, decimal
N1	=	Normal weight from logs on front two tires loaded, lb.
N2	=	Normal force on rear four tires, lb.
WR1	=	Rated tire loading capacity for a single front tire, lb.
WR2	=	Rated tire loading capacity for a single rear tire, lb.
Constant values	=	Constants from empirical research

Reference: Ashmore, et al (1985)

$$PP = N1 * (P5 - R5) + N2 * (P6 - R6)$$

PP	=	Total tractive pull parallel to ground, lb.
N1	=	Normal force on front two tires, lb.
P5	=	Gross Thrust from front two tires, lb.
R5	=	Rolling Resistance encountered by front two tires, lb.
N2	=	Normal force on rear four tires, lb.
P6	=	Gross Thrust from rear four tires, lb.
R6	=	Rolling Resistance of rear four tires , lb.

Reference: Olsen and Gibbons (1983)

$$F = PP - C * P2 - ForwarderW * Sin(T)$$

F	=	Breakout force, lb.
PP	=	Total tractive pull parallel to ground, lb.
P2	=	Tangential force of logs, lb.
ForwarderW	=	Skidder weight, lb.
T	=	Ground slope, degrees
C	=	Constat that represents the breakout force of 1.1 times the skidding resistance.

Reference: Olsen and Gibbons (1983)

Once the logs are free, then the vehicle velocity is calculated.

Velocity

$$V2 = [workpace]/[100*(1-S)*(h1*33*1000)]/[(N1*R5 + N2*R6 + P2 + ForwarderW*\sin(T))]$$

V2	=	Loaded velocity, ft/sec
workpace	=	Work pace adjustment factor, decimal
S	=	Slip, decimal
h1	=	Horsepower available at axle, hp
N1	=	Normal weight on front two tires, lb.
R5	=	Rolling resistance encountered by front two tires, lb.
N2	=	Normal weight on rear four tires, lb.
R6	=	Rolling resistance encountered by rear four tires, lb.
P2	=	Tangential force of logs, lb.
ForwarderW	=	Forwarder weight, lb.
T	=	Ground Slope, degrees

Reference: Olsen and Gibbons (1983)

Mean contact stress is calculated to estimate the minimum average ground pressure of the vehicle on the soil.

Mean Contact Stress

$$P_m = k * p_i$$

Pm	=	Mean Contact Stress, psi
k	=	Constant from imperical research
p_i	=	Tire inflation pressure, psi

Reference: Koolen and Kuipers (1983)

The *k* constant has one of four different values based on ply rating of the tire. SkidPC 2000 selects the constant value from Koolen and Kuipers that most closely matches the user input ply rating. The different values that are possible are as follows:

Ply Rating	Constant
< 6	1.10
>6 and <10	1.15
>10 and <16	1.20
>16	1.25

Reference: Koolen and Kuipers (1983)

After all the calculations for the loaded condition are completed, then the program will analyze the unloaded return trip. Travel time and productivity equations are then calculated and the results are reported the same as in the rubber tired skidder section.

RUT DEPTH EQUATIONS

To calculate the number of round trip vehicle passes before a user defined limiting rut depth is achieved, the first step is to analyze the slash located in the skid trail.

$$SlashFactor = \frac{1}{(0.0075 * \text{Slash} + 0.93)}$$

SlashFactor	=	Nominal ground pressure multiplier, decimal
Slash	=	Tons/acre of slash in skid trail,
Constant Values	=	Constants from empirical research

Reference: Wronski and Humphreys (1994) Equation converted from SI units as in reference

The next step is to figure out how the rut depth is affected by multiple passes over a soil.

$$z = \left(z_1^2 + z_2^2\right)^{1/2}$$

z_1^2	=	Initial rut depth prior to passage of wheel (squared), in.
z_2^2	=	Expected rut depth if the soil has been undisturbed (squared), in.
z	=	Cummulative or total rut depth, in.

Reference: Wronski and Humphreys (1994)

For each of the vehicle types, ground pressures are calculated and adjusted by the slash factor for calculation of the rut depth after a single wheel (or track) has passed. The equations were developed for a range of cone index values between 80 and 240 psi and slash amounts in the skid trails between 0 and 18 tons/acre.

Tracked skidder

$$\text{MMP} = \frac{(1.26 * \text{CrawlerW})}{2 * \text{NTrackWheels} * \text{TrackWd} * \sqrt{\text{TrackPitch} * \text{TrackWheelDiam}}}$$

CrawlerW	=	Tracked skidder weight, lb.
NtrackWheels	=	Number of track wheels
TrackWd	=	Track width, in.
TrackPitch	=	Track Pitch, in.

TrackWheelDiam	=	Track Wheel Diameter, in.
MMP	=	Mean Maximum Pressure, psi
Constant Values	=	Constants from empirical research

Reference: Wronski and ***Humphreys (1994)***

$$\mathrm{EGPTrack} = 0.8 * \mathrm{NGP} * \left(\frac{\mathrm{MMP}}{2 * \mathrm{NGP}}\right)^{1.23}$$

EGPTrack	=	Effective Track Ground Pressure, psi
NGP	=	Nominal Ground Pressure, psi
MMP	=	Mean Maximum Pressure, psi
Constant Values	=	Constants from empirical research

Reference: Wronski and Humphreys (1994)

$$\mathrm{RutDepth} = \frac{4.61 * \mathrm{D}}{\left(\frac{\mathrm{BCI}}{\mathrm{EGP}}\right)^{\frac{13}{5}}}$$

RutDepth	=	Rut Depth, in.
NGP	=	Effective Track Ground Pressure, psi
BCI	=	Beginning Cone Index, psi
D	=	Track Wheel Diameter, in.
Constant Values	=	Constants from empirical research

Reference: Wronski and Humphreys (1994)

<u>Rubber-tired Skidder and Forwarder/Clambunk</u>

$$NGP = \frac{DynamicWL}{(B1 * D1)}$$

NGP	=	Nominal Ground Pressure, psi
DynamicWL	=	Dynamic Wheel Load, lb.
B1	=	Tire width, in.
D1	=	Tire diameter, in.

Reference: Wronski and Humphreys (1994)

$$RutDepth = \frac{4.61 * D1}{\left(\frac{BCI}{NGP}\right)^{\frac{13}{5}}}$$

RutDepth	=	Rut Depth, in.
D1	=	Tire Diameter, in.

BCI	=	Beginning Cone Index, psi
NGP	=	Nominal Ground Pressure, psi
Constant Values	=	Constants from empirical research

Reference: Wronski and Humphreys (1994)

APPENDIX B: SELECTED CODE

TRACKED SKIDDER MODULE

```
Option Explicit
Public Sub CrawlerAnalysis()
On Error GoTo ErrorHandler

  '--- init variables ---
    t2 = 0    ' set cumlative travel time = 0
    eror = 0
    If FTrack = 0 Then NBOGIE = NTrackWheels * 2 + 4: RDIA = 0.346 * TrackLn
    If FTrack = 1 Then NBOGIE = NRoadWheels * 2: RDIA = 2 * TrackLn / NBOGIE
    If Buttfirst = 1 Then T8 = 1 Else T8 = 1.5

    frmCrawlerResults.FormatfgCrawlerResults

  '--- iterate segments ---
For j = 0 To SEGMENTS - 1
    If SoilType(j) = 1 Then
        n(j) = 0.4: KC(j) = 17: K0(j) = 12        'bekker soil parameters - fine grain
        Else: n(j) = 0.66: KC(j) = 20: K0(j) = 16 'bekker soil parameters - coarse grain
        'KC = cohesive modules of deformation, N exponet of deformation
        'K0 = frictional modulus of deformation
    End If

    t1 = Slope(j)
    A2 = Distance(j)
    RCI = CI(j)
    T = Atn(t1 / 100)



'-----------output basic data
    Dim entry As String
    frmCrawlerResults.fgCrawlerResults.TextMatrix(1, j + 2) = Format(Slope(j) / 100, "percent")

    If METRIC = False Then
        frmCrawlerResults.fgCrawlerResults.TextMatrix(2, j + 2) = Distance(j)
        frmCrawlerResults.fgCrawlerResults.TextMatrix(3, j + 2) = CI(j)
    Else
        frmCrawlerResults.fgCrawlerResults.TextMatrix(2, j + 2) = Format((Distance(j) * 0.3048), "fixed")  'convert
feet to meters
        frmCrawlerResults.fgCrawlerResults.TextMatrix(3, j + 2) = Format((CI(j) * 6.89475729317), "fixed") ' convert
psi to kP
    End If

    If frmCrawlerProfile.txtSoilType(j).text = 1 Then
        entry = "fine"
        Else: entry = "coarse"
    End If
    frmCrawlerResults.fgCrawlerResults.TextMatrix(4, j + 2) = entry
            If frmCrawlerProfile.txtPasses(j).text = 0 Then
                entry = "single"
                Else: entry = "multiple"
            End If
    frmCrawlerResults.fgCrawlerResults.TextMatrix(5, j + 2) = entry

'--- UNLOADED DIRECTION ---
    TRIP = 0
    T = T * (-1)
    P1 = 0
    P2 = 0

    PressureDistribution
    ComputeVelocity

    If eror = 1 Then GoTo 3113  'error trap to end of analysis

    V3 = VA * 88 * Workpace / 100 '--- UNLOADED VELOCITY V3 (FT./MIN.) ---
    S1 = SLIPX

'--- LOADED DIRECTION ---
    TRIP = 1
    T = T * (-1)
    If TurnWt = 0 Then
            P1 = 0
            P2 = 0
        Else
            LogForces
    End If
    PressureDistribution
    ComputeVelocity

    If eror = 1 Then GoTo 3113

    V2 = VA * 88 * (Workpace / 100) '--- LOADED VELOCITY V2 (FT./MIN.) ---
    S = SLIPX
'--- CUMULATIVE TRAVEL TIME T2 ---
    If (V2 = 0 Or V3 = 0) Then t2 = 0 Else t2 = A2 / V2 + A2 / V3 + t2

    '=========================================================================
    '=  send results for this road segment to frmCrawlerResults rich text box  =
    '=========================================================================
    fgCrawlerResults

Next j

'--- ADD NON-TRAVEL TIME TO THE CUMULATIVE TRAVEL TIME ---
    t2 = t2 + T3
    If METRIC = True Then
        LBSPERHOUR = 60 * (TurnWt * 0.45359237) / t2 * (100 - Delay) / 100         'kg PER HOUR
    Else
        LBSPERHOUR = 60 * TurnWt / t2 * (100 - Delay) / 100         'LBS PER HOUR
    End If
'=========================================================================
'=  send results for total production summay to frmCrawlerResults label   =
'=========================================================================
If METRIC = False Then
    frmCrawlerResults.lblCrawlerProduction(5).Caption = Format(TurnWt, "fixed")
Else
    frmCrawlerResults.lblCrawlerProduction(5).Caption = Format(TurnWt * 0.45359237, "fixed")
End If

    frmCrawlerResults.lblCrawlerProduction(6).Caption = Format(t2, "fixed")
    frmCrawlerResults.lblCrawlerProduction(7).Caption = Format(LBSPERHOUR / LbsMbf, "fixed")
    frmCrawlerResults.lblCrawlerProduction(8).Caption = Format(LBSPERHOUR / LbsCunit, "fixed")
    frmCrawlerResults.lblCrawlerProduction(9).Caption = Format(F2 / LBSPERHOUR * LbsMbf, "Currency")
    frmCrawlerResults.lblCrawlerProduction(10).Caption = Format(F2 / LBSPERHOUR * LbsCunit, "Currency")

3113
If eror = 1 Then
    Unload frmCrawlerSpecs
    Load frmCrawlerMenu
    frmCrawlerMenu.Visible = True
    frmCrawlerResults.Visible = False
    Else
End If

Exit Sub

ErrorHandler:
GlobalErrorHandler

End Sub
Public Sub LogForces()
'On Error GoTo ErrorHandler
  If Arch = 1 Then
    H = Yp / 12 - 1.4     'distance from ground to hook point (measured perpendicular to ground)
    l1 = 1                'distance along log from leading end to chocker hook point
    l2 = 0.2 * l3         'average lenght of contact of log and ground (perumpral's log model)
    C8 = 0.4 * l3 * T8    'distance from leading end of log to center of gravity
    x = l3 - l1 - l2      'suspended log lenght (choker attachment point to ground contact point)
    q1 = (1 - (H / x) ^ 2) ^ 0.5   'Q1,Q2,Q3,Q4 = algebraic subs in Perumpral's log model as mod by Gibbons and Olsen
    q2 = Sin(T) * H * ((l3 - l2 - C8) / x - 1)
    q3 = q1 * (C8 - l1) * Cos(T)
    q4 = u * H + l2 / 2 + x * q1
    q = TurnWt * (q3 + q2) / q4   'normal componet of the portion of the log weight transfered to the ground
    P1 = TurnWt * Cos(T) - q    'componet of winchline tension perpendicular to ground
    P2 = TurnWt * Sin(T) + u * q  'componet of winchline tension parallel to ground
  Else
    N7 = 0.2        'portion of log wight transferred to vehicle (herrick's and Fiske and Fridley's log models)
    C7 = 0.9 + 1.667 * Tan(T) 'coefficient of resistance to skidding (herrick's and Fiske and Fridley's log models)
    P1 = N7 * TurnWt * Cos(T)
    P2 = (1 - N7) * C7 * TurnWt * Cos(T) + TurnWt * Sin(T)
    q = (1 - N7) * TurnWt * Cos(T)
  End If
'--- WINCHLINE TENSION DURING BREAKOUT = MAXTENS ---
  MAXTENS = Sqr((1.1 * (P2 - TurnWt * Sin(T)) + TurnWt * Sin(T)) ^ 2 + P1 ^ 2)

'ErrorHandler:
'GlobalErrorHandler

End Sub
Public Sub PressureDistribution()
'On Error GoTo ErrorHandler
  GCW = CrawlerW * Cos(T) + P1
  XR = (CrawlerW * Xc * Cos(T) + CrawlerW * Yc * Sin(T) + P1 * Xp + P2 * Yp) / GCW
  If (XR / TrackLn) < (1 / 3) Then
'--- PRESSURE DISTRIBUTION CASE 1 ---
```

```
    PR = 0
    PF = GCW / (3 * TrackWd * XR)
    LCL = 3 * XR
  ElseIf (XR / TrackLn) <= (2 / 3) Then
'--- PRESSURE DISTRIBUTION CASE 2 ---
    PR = (3 * GCW / (TrackWd * TrackLn)) * (XR / TrackLn - 1 / 3)
    PF = (3 * GCW / (TrackWd * TrackLn)) * (2 / 3 - XR / TrackLn)
    LCL = TrackLn
  Else
'--- PRESSURE DISTRIBUTION CASE 3 ---
    PF = 0
    PR = GCW / (3 * TrackWd * TrackLn * (1 - XR / TrackLn))
    LCL = TrackLn - 3 * TrackLn * (XR / TrackLn - 2 / 3)
  End If
'--- TEST IF CONTACT LENGTH IS ZERO OR NEGATIVE ---
  If LCL <= 0 Then
    MsgBox "ERROR: VEHICLE IS IMMOBILIZED" & vbCrLf & "CHECK PROFILE SEGMENT # " & j & vbCrLf & "(try modifying machine
specs or conditions)"
    eror = 1
    frmCrawlerResults.ClearCrawlerResultsVariables
    Exit Sub
    End If
  CPAVG = GCW / (2 * TrackWd * TrackLn) 'CPAVG = AVG. PRESSURE UNDER TRACKS
  KCBK0 = KC(j) / TrackWd + K0(j)       'factor which is a function of soil sinkage (KC and KO)and track width
'***DIVIDE TRACK UP INTO SEGMENTS***
  SEGS = NBOGIE / 2 - 1 'SEGS = NO. OF SPANS BETWEEN ROLLERS, ONE TRACK
  BCK = SEGS + 1        'BCK  = NO. OF REACTION POINTS, ONE TRACK
  If SEGS <= 1 Then
    AINC(1) = TrackLn * TrackWd 'AINC=AREA FOR REACTION POINT, BOTH TRACKS
    AINC(2) = AINC(1)
    XX(1) = 0                   'XX=DISTANCE ALONG TRACK FROM FRONT IDLER TO REACTION POINT
    XX(2) = TrackLn
  ElseIf SEGS <= 2 Then
    XX(1) = 0
    XX(2) = TrackLn / 2
    XX(3) = TrackLn
    AINC(1) = TrackLn * TrackWd / 2
    AINC(2) = 2 * AINC(1)
    AINC(3) = AINC(1)
  Else
    SPROC = RDIA / 2 + (TrackLn - RDIA) / (SEGS - 1) / 2 'rigid track only - dist from center of front/rear track wheel
to adjacent roller
    ROLL = (TrackLn - SPROC * 2) / (SEGS - 2)            'Roll = dist along tracks
    For inc = 3 To (BCK - 2): AINC(inc) = TrackWd * ROLL * 2: Next inc
    AINC(1) = (SPROC / 2) * TrackWd * 2
    AINC(2) = (SPROC + ROLL) * TrackWd
    AINC(BCK - 1) = AINC(2)
    AINC(BCK) = AINC(1)
    XX(1) = 0
    XX(2) = SPROC
    XX(BCK - 1) = TrackLn - SPROC
    XX(BCK) = TrackLn
    For inc = 3 To (BCK - 2): XX(inc) = SPROC + ROLL * (inc - 2): Next inc
  End If

'--- ITERATE TO FIND REACTION POINTS AT EACH END OF THE TRACKS ---
  PRODOLD = 0
  If PF <= PR Then
'***PR>PF**************************************
     RBCKMIN = ((PR + PF) / 2 * 1.28 * TrackLn / LCL - 1) * AINC(BCK)
     RBCKMAX = RBCKMIN + 2000
     R1MAX = PR * AINC(BCK)
     UP = -1000
     R1 = R1MAX
     Flag = 0
7510 RBCK = RBCKMAX  'RBCK=point of reaction of ground under rear wheel of track negative if not on ground
     SumForcesNormalToGround
     RBCK1 = RBCK
     SUM1 = SUM
     RBCK = RBCKMIN
     SumForcesNormalToGround
     RBCK2 = RBCK
     SUM2 = SUM
     CNTR = 1   'cntr = var used to count # of secant methods iterations
7600 M = (SUM2 - SUM1) / (RBCK2 - RBCK1)
     RBCK1 = RBCK2
     SUM1 = SUM   'sum = sum of static forces perpendicular to ground
     RBCK2 = RBCK2 - SUM2 / M
     RBCK = RBCK2
     SumForcesNormalToGround
     SUM2 = SUM
     If Abs(SUM) <= 1 Then GoTo 7820
     CNTR = CNTR + 1
     If CNTR < 25 Then GoTo 7600 Else GoTo 7700
7700 SUMOLD = 0
     UP1 = -1000
     RBCK = PR * AINC(BCK) * 10
7730 SumForcesNormalToGround
```

```
     If SUM * SUMOLD < 0 Then GoTo 7780
     SUMOLD = SUM
     RBCK = RBCK + UP1
     GoTo 7730
7780 If Abs(SUM) < 10 Then GoTo 7820
     RBCK = RBCK - UP1
     UP1 = UP1 / 2
     GoTo 7730
7820 SumMomentsAboutFrontWheels
     If Flag = 1 Then GoTo 8440
     If PROD * PRODOLD < 0 Then GoTo 7880  'PROD = sum of the moments about leading edge of tracks
     PRODOLD = PROD
     R1 = R1 + UP
     GoTo 7510
7880 If Abs(PROD) < 100 Then GoTo 8440
     If Abs(PRODOLD) < 100 And PRODOLD <> 0 Then Flag = 1
     R1 = R1 - UP
     UP = UP / 2
     GoTo 7510
  Else  '--- PF > PR ---
     R1MIN = ((PR + PF) / 2 * 1.28 * TrackLn / LCL - 1) * AINC(1)
     R1MAX = R1MIN + 2000
     RBCKMAX = PF * AINC(1)
     UP = -1000
     RBCK = RBCKMAX
     Flag = 0
8000 R1 = R1MAX
     SumForcesNormalToGround
     R11 = R1
     SUM1 = SUM
     R1 = R1MIN
     SumForcesNormalToGround
     R12 = R1
     SUM2 = SUM
     CNTR = 1
8090 M = (SUM2 - SUM1) / (R12 - R11)   ' M=Slope of a function (used in secant method)
     R11 = R12
     SUM1 = SUM
     R12 = R12 - SUM2 / M
     R1 = R12
     SumForcesNormalToGround
     SUM2 = SUM
     If Abs(SUM) <= 1 Then GoTo 8310
     CNTR = CNTR + 1
     If CNTR < 25 Then GoTo 8090 Else GoTo 8190
8190 SUMOLD = 0
     UP1 = -1000
     R1 = PF * AINC(1) * 10
8220 SumForcesNormalToGround
     If SUM * SUMOLD < 0 Then GoTo 8270
     SUMOLD = SUM
     R1 = R1 + UP1
     GoTo 8220
8270 If Abs(SUM) < 10 Then GoTo 8310
     R1 = R1 - UP1
     UP1 = UP1 / 2
     GoTo 8220
8310 SumMomentsAboutFrontWheels
     If Flag = 1 Then GoTo 8440
     If PROD * PRODOLD < 0 Then GoTo 8370
     PRODOLD = PROD
     RBCK = RBCK + UP
     GoTo 8000
8370 If Abs(PROD) < 100 Then GoTo 8440
     If Abs(PRODOLD) < 100 And PRODOLD <> 0 Then Flag = 1
     RBCK = RBCK - UP
     UP = UP / 2
     GoTo 8000
     '
  End If
8440 '==========================< END ITERATION >=============================
  r(1) = R1
  r(BCK) = RBCK
  If r(1) >= r(BCK) Then PMAX = r(1) / AINC(1)
  If r(BCK) > r(1) Then PMAX = r(BCK) / AINC(BCK)
'  Print #1,
    If TRIP = 0 Then 'unloaded
        If METRIC = False Then
            frmCrawlerResults.fgCrawlerResults.TextMatrix(6, j + 2) = "Unloaded Direction"
            frmCrawlerResults.fgCrawlerResults.TextMatrix(7, j + 2) = Format(XR, "fixed")
            frmCrawlerResults.fgCrawlerResults.TextMatrix(8, j + 2) = Format(LCL, "fixed")
            frmCrawlerResults.fgCrawlerResults.TextMatrix(9, j + 2) = Format(PMAX, "fixed")
        Else
            frmCrawlerResults.fgCrawlerResults.TextMatrix(6, j + 2) = "Unloaded Direction"
            frmCrawlerResults.fgCrawlerResults.TextMatrix(7, j + 2) = Format(XR * 2.54, "fixed")  'converted in to cm
            frmCrawlerResults.fgCrawlerResults.TextMatrix(8, j + 2) = Format(LCL * 2.54, "fixed") 'converted in to cm
            frmCrawlerResults.fgCrawlerResults.TextMatrix(9, j + 2) = Format(PMAX * 6.89475729317, "fixed") 'converted
psi to kPa
```

```
        End If  'end conversion to metric for unloaded
    Else  'loaded
        If METRIC = False Then
            frmCrawlerResults.fgCrawlerResults.TextMatrix(10, j + 2) = "Loaded Direction"
            frmCrawlerResults.fgCrawlerResults.TextMatrix(11, j + 2) = Format(XR, "fixed")
            frmCrawlerResults.fgCrawlerResults.TextMatrix(12, j + 2) = Format(LCL, "fixed")
            frmCrawlerResults.fgCrawlerResults.TextMatrix(13, j + 2) = Format(PMAX, "fixed")
        Else 'METRIC  = true
            frmCrawlerResults.fgCrawlerResults.TextMatrix(10, j + 2) = "Loaded Direction"
            frmCrawlerResults.fgCrawlerResults.TextMatrix(11, j + 2) = Format(XR * 2.54, "fixed") 'converted in to cm
            frmCrawlerResults.fgCrawlerResults.TextMatrix(12, j + 2) = Format(LCL * 2.54, "fixed") 'converted in to cm
            frmCrawlerResults.fgCrawlerResults.TextMatrix(13, j + 2) = Format(PMAX * 6.89475729317, "fixed") 'converted
psi to kPa
        End If   'end conversion to metric for loaded
    End If

  '--- FIND INTERMEDIATE REACTION POINTS ---
  ZZ(1) = (Abs(R1) / (AINC(1) * KCBK0)) ^ (1 / n(j))
  If R1 < 0 Then ZZ(1) = ZZ(1) * (-1): r(1) = 0
  ZZ(BCK) = (Abs(RBCK) / (AINC(BCK) * KCBK0)) ^ (1 / n(j))
  If RBCK < 0 Then ZZ(BCK) = ZZ(BCK) * (-1): r(BCK) = 0
  If ZZ(1) * ZZ(BCK) >= 0 Then
    LCL = TrackLn
  Else
    If ZZ(1) < ZZ(BCK) Then Z = ZZ(BCK)
    If ZZ(1) >= ZZ(BCK) Then Z = ZZ(1)
    LCL = Z / ((Abs(ZZ(1)) + Abs(ZZ(BCK))) / TrackLn)
  End If
  For inc = 2 To (BCK - 1)
    r(inc) = 0: ZZ(inc) = ((ZZ(BCK) - ZZ(1)) / TrackLn) * XX(inc) + ZZ(1)
    If ZZ(inc) > 0 Then r(inc) = ZZ(inc) ^ n(j) * AINC(inc) * KCBK0
  Next inc
  RCNU = 0
  For inc = 1 To BCK
    P(inc) = r(inc) / AINC(inc)
    RCNU = RCNU + 2 * TrackWd * KCBK0 * (P(inc) / KCBK0) ^ ((n(j) + 1) / n(j)) / (n(j) + 1) * AINC(inc) / (2 * TrackWd)
/ TrackLn
  Next inc
  RCUN = 2 * TrackWd * KCBK0 * (CPAVG / KCBK0) ^ ((n(j) + 1) / n(j)) / (n(j) + 1)
  CRINC = RCNU / RCUN

'ErrorHandler:
'GlobalErrorHandler: Resume

End Sub
Public Sub ComputeVelocity() '--- COMPUTE VELOCITY USING NATO MODEL ALGORITHMS ---
'On Error GoTo ErrorHandler

  GCW = CrawlerW * Cos(T) + P1 '--- GROSS COMBINED WEIGHT --- 'P1 = componet of winchline parallel to ground
  HPT = (HP * Efficiency) / (GCW / 2000) '--- HORSEPOWER PER TON OF GCW ---
  '--- WEIGHT FACTOR ---
  WF = 1
  If GCW / 1000 > 50 Then WF = 1.2
  If GCW / 1000 > 70 Then WF = 1.4
  If GCW / 1000 > 100 Then WF = 1.8
  TF = TrackWd / 100 '--- TRACK FACTOR ---
  GF = 1.1 '--- GROUSER FACTOR ---
  If GH < 1.5 Then GF = 1
  EF = 1: If HPT < 10 Then EF = 1.05 '--- ENGINE FACTOR ---

  '--- TRANSMISSION FACTOR ---
  If TQ = 1 Then TFX = 1     'if machine has powershift transmission
  If TQ = 0 Then TFX = 1.05  'if machine doesn't powershift transmission('if machine doesn't have torque converter)

  CPAVG = GCW / (2 * TrackWd * TrackLn) '--- AVERAGE PRESSURE ON TRACKS ---

  If SoilType(j) = 1 Then '--- FINE-GRAINED SOIL ---
    '---BOGIE LOAD RANGE FACTOR---
    WLORF = GCW / 10 / NBOGIE / AShoe
    '---CLEARANCE FACTOR---
    CLF = (ClrMin / 10)
    '---MOBILITY INDEX---
    XMI = (CPAVG * WF / TF / GF + WLORF - CLF) * EF * TFX
    '---VEHICLE CONE INDEX FOR FINE-GRAINED SOIL---
    ' VCI FOR ONE PASS
    If passes(j) = 0 Then VCIFG = 7 + 0.2 * XMI - 39.2 / (XMI + 5.6)
    ' VCI FOR 50 PASSES
    If passes(j) = 1 Then VCIFG = 19.27 + 0.43 * XMI - 125.79 / (XMI + 7.08)
    '---EXCESS CONE INDEX FOR FINE-GRAINED SOIL---
    RCIX = RCI - VCIFG
    '---DRAWBAR PULL COEFFICIENT DOWPB FOR FINE-GRAINED SOIL---
    If CPAVG < 4 Then
      If RCIX < 0 Then
        DOWPB = 0.076 * RCIX
      Else
        DOWPB = 0.544 + 0.0463 * RCIX - ((0.544 + 0.0463 * RCIX) ^ 2 - 0.07020001 * RCIX) ^ 0.5
      End If
    Else
```

```
      If RCIX < 0 Then
        DOWPB = 0.056 * RCIX
      Else
        DOWPB = 0.4554 + 0.0392 * RCIX - ((0.4554 + 0.0392 * RCIX) ^ 2 - 0.0526 * RCIX) ^ 0.5
      End If
    End If
    '--- MOTION RESISTANCE COEFFICIENT RTOWPB FOR FINE-GRAINED SOIL ---
    If RCIX < 0 Then
      If CPAVG < 4 Then RTOWPB = 0.4 - 0.072 * RCIX Else RTOWPB = 0.4 - 0.052 * RCIX
    Else
      RTOWPB = 0.045 + 2.3075 / (RCIX + 6.5)
    End If
    '--- F-G SOIL CORRECTION FACTOR, AND SOIL LIMITED TRACTIVE EFFORT ---
    If CPAVG < 4 Then CF = DOWPB - 0.758 + RTOWPB: TFOR = (CF + 0.82) * GCW
    If CPAVG >= 4 Then CF = DOWPB - 0.671 + RTOWPB: TFOR = (CF + 0.71) * GCW
  Else
    '--- COARSE-GRAINED SOIL ---
    '---DRAWBAR PULL COEFFICIENT DOWPB FOR COARSE-GRAINED SOIL---
    G = RCI * 0.8645 / 3
    If FTrack = 0 Then PIT = 0.63 * G * ((TrackWd * TrackLn) ^ 1.5) / (GCW / 2) 'FLEXIBLE TRACK
    If FTrack = 1 Then PIT = 1 * G * ((TrackWd * TrackLn) ^ 1.5) / (GCW / 2) 'RIGID TRACK
    If PIT <= 25 Then
      DOWPB = 0.121 + 0.258 * LOG(PIT) / 2.302
    ElseIf PIT <= 100 Then
      DOWPB = 0.339 + 0.109 * LOG(PIT) / 2.302
    ElseIf PIT <= 1000 Then
      DOWPB = 0.481 + 0.038 * LOG(PIT) / 2.302
    Else: DOWPB = 0.595
    End If
    '---MOTION RESISTANCE COEFFICIENT FOR COARSE-GRAINED SOIL---
    RTOWPB = 0.6 - DOWPB + 0.045
    '--- C-G SOIL CORRECTION FACTOR, AND SOIL LIMITED TRACTIVE EFFORT ---
    If FTrack = 1 Then CF = 0.074: TFOR = (CF + 0.568) * GCW
    If FTrack = 0 Then CF = 0.1: TFOR = (CF + 0.695) * GCW
  End If
  '
  '---GRADE RESISTANCE---
  FGRADE = CrawlerW * Sin(T)
  '---PAYLOAD RESISTANCE=P2---
  '---COMPACTION RESISTANCE CR---
  SURFF = 1  'surface roughness factor = 1
  RS = (GCW / 2000) * (30 + 0.9 * 4)        'internal resistnace-mainly frictional losses
  CR = SURFF * (RTOWPB * GCW - RS) * CRINC  'factor used to increase the compaction resistance from
NATO***************************
  '---SUM OF RESISTANCES---
  RESIST = CR + FGRADE + P2
  If RESIST < 0 Then RESIST = 0
  '---THEORETICAL VELOCITY VT ---
  TheoreticalVelocity
  RS = (GCW / 2000) * (30 + 0.9 * VT)
  CR = SURFF * (RTOWPB * GCW - RS) * CRINC
  '
  '---SUM OF RESISTANCES---
  RESIST = CR + FGRADE + P2: If RESIST < 0 Then RESIST = 0
  '---THEORETICAL VELOCITY VT ---
  TheoreticalVelocity
  '--- TEST IF ENOUGH SOIL TRACTIVE EFFORT TO OVERCOME RESISTANCES ---
    If TFOR <= (CR + FGRADE + P2 + RS) Then
        MsgBox "WARNING: THE RESISTANCES EXCEED THE MAXIMUM TRACTIVE EFFORT THAT THE TRACKS CAN DEVELOP." & vbCrLf &
"CHECK PROFILE SEGMENT # " & j & vbCrLf & "(try changing soil conditions)"
        eror = 1
        frmCrawlerResults.ClearCrawlerResultsVariables
        Exit Sub
    End If

    If (TFOR - CR - FGRADE - RS) <= (1.1 * (P2 - TurnWt * Sin(T)) + TurnWt * Sin(T)) Then
        MsgBox "WARNING: THE BREAKOUT RESISTANCES EXCEED THE MAXIMUM TRACTIVE EFFORT  THAT THE TRACKS CAN DEVELOP." _
            & vbCrLf & "CHECK PROFILE SEGMENT # " & j & vbCrLf & "(try modifying machine specs or conditions)"
        eror = 1
        frmCrawlerResults.ClearCrawlerResultsVariables
        Exit Sub
    End If
  '--- CALCULATE SLIP OF TRACKS ON GROUND ---
    If (CR + FGRADE + P2 + RS) > 0 Then YX = (CR + FGRADE + P2 + RS) / GCW - CF 'YX= corrected drawbar pull coefficient
    If (CR + FGRADE + P2 + RS) <= 0 Then YX = -CF

    If SoilType(j) = 1 Then

    '--- SLIP FOR FINE-GRAINED SOIL ---
    If (CPAVG < 4 And YX >= 0.71) Or (CPAVG >= 4 And YX >= 0.82) Then
        MsgBox "ERROR: TRACKS SPIN OUT AT 100% SLIP" & vbCrLf & "CHECK PROFILE SEGMENT # " & j & vbCrLf & "(try
modifying machine specs or conditions)"
        eror = 1
        frmCrawlerResults.ClearCrawlerResultsVariables
        Exit Sub
    End If

    If CPAVG < 4 Then SLIPX = 0.0257 * YX - 0.0161 + 0.01519 / (0.8353 - YX)
    If CPAVG >= 4 Then SLIPX = 0.0733 * YX - 0.0063 + 0.00734 / (0.7177 - YX)
    If SLIPX < -0.01 Or SLIPX > 1 Then
        MsgBox "ERROR: SLIP IS NEGATIVE OR GREATER THAN 100%" & vbCrLf & "CHECK PROFILE SEGMENT # " & j & vbCrLf &
"(try modifying machine specs or conditions)"
        eror = 1
        frmCrawlerResults.ClearCrawlerResultsVariables
        Exit Sub
    End If
    SLIPX = Abs(SLIPX)
    VA = VT * (1 - SLIPX)
  Else
    '--- SLIP FOR COARSE-GRAINED SOIL ---
    If FTrack = 0 Then
      '--- RIGID TRACK ---
    If YX >= 0.5677 Then
        MsgBox "ERROR: TRACKS SPIN OUT AT 100%" _
        & vbCrLf & "CHECK PROFILE SEGMENT # " & j & vbCrLf & "(try modifying machine specs or conditions)"
        eror = 1
        frmCrawlerResults.ClearCrawlerResultsVariables
        Exit Sub
    End If
      SLIPX = -0.0083 + 0.005312 / (0.573 - YX)
      If SLIPX < -0.01 Or SLIPX > 1 Then
        MsgBox "ERROR: SLIP IS NEGATIVE OR GREATER THAN 100%" _
        & vbCrLf & "CHECK PROFILE SEGMENT # " & j & vbCrLf & "(try modifying machine specs or conditions)"
        eror = 1
        frmCrawlerResults.ClearCrawlerResultsVariables
        Exit Sub
      End If
      SLIPX = Abs(SLIPX)
      VA = VT * (1 - SLIPX)
    Else
      '--- FLEXIBLE TRACK ---
    If YX >= 0.695 Then
        MsgBox "ERROR: TRACKS SPIN OUT AT 100% SLIP" & vbCrLf & "CHECK PROFILE SEGMENT # " & j & vbCrLf & "(try
modifying machine specs or conditions)"
        eror = 1
        frmCrawlerResults.ClearCrawlerResultsVariables
        Exit Sub
    End If
      YY = 1.704 * YX - 0.72
      SLIPX = YY + ((YY ^ 2) + 0.09000001 * YX + 0.008999999) ^ 0.5
      If SLIPX < -0.01 Or SLIPX > 1 Then
        MsgBox "ERROR: SLIP IS NEGATIVE OR GREATER THAN 100%" _
        & vbCrLf & "CHECK PROFILE SEGMENT # " & j & vbCrLf & "(try modifying machine specs or conditions)"
        eror = 1
        frmCrawlerResults.ClearCrawlerResultsVariables
        Exit Sub
      End If
      SLIPX = Abs(SLIPX)
      VA = VT * (1 - SLIPX)
      'Return   '**************************taken out --don't think there is gosub
    End If
  End If 'coarse-grained soil

'ErrorHandler:
'GlobalErrorHandler: Resume

End Sub
Public Sub TheoreticalVelocity()
'On Error GoTo ErrorHandler

'--- SUBROUTINE TO GET THEOR VELOC FROM DRAWBAR PULL VS SPEED CURVES ---
    If RESIST > DP(1) Then
        MsgBox "ERROR: VEHICLE DOES NOT HAVE ENOUGH POWER TO OVERCOME RESISTING FORCES" _
        & vbCrLf & "CHECK PROFILE SEGMENT # " & j & vbCrLf & "(try modifying machine specs or conditions)"
        eror = 1
        frmCrawlerResults.ClearCrawlerResultsVariables
        Exit Sub
    End If

    If (DP(1) - CR - FGRADE) <= (1.1 * (P2 - TurnWt * Sin(T)) + TurnWt * Sin(T)) Then
        MsgBox "ERROR: NOT ENOUGH VEHICLE POWER AVAILABLE TO OVERCOME BREAKOUT FORCE" _
        & vbCrLf & "CHECK PROFILE SEGMENT # " & j & vbCrLf & "(try modifying machine specs or conditions)"
        eror = 1
        frmCrawlerResults.ClearCrawlerResultsVariables
        Exit Sub
    End If

'--- compute Velocity Theroetical (VT)
NGEAR = 1
Do While (DP(NGEAR) <> 0 Or V(NGEAR) <> 0) And NGEAR <= 6
    If RESIST >= DP(NGEAR + 1) Then
        VT = V(NGEAR) + (V(NGEAR + 1) - V(NGEAR)) / (DP(NGEAR) - DP(NGEAR + 1)) * (DP(NGEAR) - RESIST)
        GoTo 1313
    End If
    NGEAR = NGEAR + 1
Loop
```

```
1313

'ErrorHandler:
'GlobalErrorHandler: Resume

End Sub
Public Sub SumForcesNormalToGround()
'On Error GoTo ErrorHandler
  r(1) = R1
  r(BCK) = RBCK
  ZZ(1) = (Abs(R1) / (AINC(1) * KCBK0)) ^ (1 / n(j))
  If R1 < 0 Then ZZ(1) = ZZ(1) * (-1): r(1) = 0
  ZZ(BCK) = (Abs(RBCK) / (AINC(BCK) * KCBK0)) ^ (1 / n(j))
  If RBCK < 0 Then ZZ(BCK) = ZZ(BCK) * (-1): r(BCK) = 0
  SUM = CrawlerW * Cos(T) + P1 - r(1) - r(BCK)
  For inc = 2 To (BCK - 1)
    ZZ(inc) = ((ZZ(BCK) - ZZ(1)) / TrackLn) * XX(inc) + ZZ(1)
    If ZZ(inc) > 0 Then r(inc) = ZZ(inc) ^ n(j) * AINC(inc) * KCBK0: SUM = SUM - r(inc)
  Next inc

'ErrorHandler:
'GlobalErrorHandler: Resume

End Sub
Public Sub SumMomentsAboutFrontWheels()
'On Error GoTo ErrorHandler
  r(1) = R1
  r(BCK) = RBCK
  ZZ(1) = (Abs(R1) / (AINC(1) * KCBK0)) ^ (1 / n(j))
  If R1 < 0 Then ZZ(1) = ZZ(1) * (-1): r(1) = 0
  ZZ(BCK) = (Abs(RBCK) / (AINC(BCK) * KCBK0)) ^ (1 / n(j))
  If RBCK < 0 Then ZZ(BCK) = ZZ(BCK) * (-1): r(BCK) = 0
  PROD = -CrawlerW * Cos(T) * Xc - CrawlerW * Sin(T) * Yc - P1 * Xp + r(1) * XX(1) - P2 * Yp + r(BCK) * XX(BCK)
  For inc = 2 To (BCK - 1)
    ZZ(inc) = ((ZZ(BCK) - ZZ(1)) / TrackLn) * XX(inc) + ZZ(1)
    If ZZ(inc) > 0 Then r(inc) = ZZ(inc) ^ n(j) * AINC(inc) * KCBK0: PROD = PROD + r(inc) * XX(inc)
  Next inc

'ErrorHandler:
'GlobalErrorHandler: Resume

End Sub
Public Sub fgCrawlerResults() 'populate fgCrawlerResults()
'On Error GoTo ErrorHandler

    Dim entry As String
    If METRIC = False Then
    frmCrawlerResults.fgCrawlerResults.TextMatrix(14, j + 2) = Format(q, "fixed")
    frmCrawlerResults.fgCrawlerResults.TextMatrix(15, j + 2) = Format(P2, "fixed")
    frmCrawlerResults.fgCrawlerResults.TextMatrix(16, j + 2) = Format(MAXTENS, "fixed")
    frmCrawlerResults.fgCrawlerResults.TextMatrix(17, j + 2) = Format(V3 / 88, "fixed") 'v3 converted back to mi/h
    frmCrawlerResults.fgCrawlerResults.TextMatrix(18, j + 2) = Format(V2 / 88, "fixed") 'v2 converted back to mi/h
    frmCrawlerResults.fgCrawlerResults.TextMatrix(19, j + 2) = Format(S, "percent")
    frmCrawlerResults.fgCrawlerResults.TextMatrix(20, j + 2) = Format(S1, "percent")
    Else 'METRIC  = true
    frmCrawlerResults.fgCrawlerResults.TextMatrix(14, j + 2) = Format(q * 0.00444822161526, "fixed") 'converted from lb
force to kN
    frmCrawlerResults.fgCrawlerResults.TextMatrix(15, j + 2) = Format(P2 * 0.00444822161526, "fixed") 'converted from
lb force to kN,
    frmCrawlerResults.fgCrawlerResults.TextMatrix(16, j + 2) = Format(MAXTENS * 0.00444822161526, "fixed") 'converted
from lb force to kN
    frmCrawlerResults.fgCrawlerResults.TextMatrix(17, j + 2) = Format(V3 / 88 * 1.609344, "fixed") 'mi/h converted to
km/h
    frmCrawlerResults.fgCrawlerResults.TextMatrix(18, j + 2) = Format(V2 / 88 * 1.609344, "fixed") 'mi/h converted to
km/h
    frmCrawlerResults.fgCrawlerResults.TextMatrix(19, j + 2) = Format(S, "percent")
    frmCrawlerResults.fgCrawlerResults.TextMatrix(20, j + 2) = Format(S1, "percent")
    End If
'ErrorHandler:
'GlobalErrorHandler: Resume

End Sub

    frmCrawlerResults.fgCrawlerResults.TextMatrix(20, j + 2) = Format(S1, "percent")
    End If
'ErrorHandler:
'GlobalErrorHandler: Resume

End Sub
```

RUBBER-TIRED SKIDDER MODULE

```
Option Explicit
Public Sub SkidderAnalysis()

    eror = 0
    If MinSpeed = 0 Then MinSpeed = 1.5
    If MaxSpeed = 0 Then MaxSpeed = 18
    If Workpace = 0 Then Workpace = 100

    If u = 0 Then u = 0.6

    V4 = MinSpeed * 88  'convert mi/h speed to fps
    S3 = MaxSpeed * 88  'convert mi/h speed to fps

    If Buttfirst = 1 Then T8 = 1 Else T8 = 1.5

    'h1 = 0.875 * HP: If TQ = 1 Then h1 = h1 * 0.9 '*.9 FOR TORQUE CONVERTER (axle horsepower)
    h1 = (Efficiency / 100) * HP
'--- compute rated tire loads ---
    WR1 = 2.2046 * (0.75 * 0.00002735 * (6.894 * IP1) ^ 0.585 * (25.4 * B1) ^ 1.39 * (RimDiameter + B1) * 25.4)
    WR2 = 2.2046 * (0.75 * 0.00002735 * (6.894 * IP2) ^ 0.585 * (25.4 * B1) ^ 1.39 * (RimDiameter + B1) * 25.4)

'--- ANALYSIS ---------------------------

'--- init variables ---
    i = SEGMENTS '# of segments
    t2 = 0: S = 0
'--- start analysis ---
For j = 0 To i - 1

    t1 = Slope(j)         ' must be percent slope
    A2 = Distance(j)      ' must be slope distance
    C1 = CI(j)            ' must be cone index
    T = Atn(t1 / 100)     ' slope beta

    If u = 99 Then
        P1 = TurnWt * Cos(T) 'normal force of logs
        P2 = TurnWt * Sin(T) 'tangential force of logs
        GoTo skipfriction    'go to subroutine to avoid log friction
    End If
'--- output basic data
    If TurnWt > 10 Then
        If METRIC = False Then
        frmSkidderResults.fgSkidderResults.TextMatrix(1, j + 2) = Format(frmSkidderProfile.txtSlope(j).text / 100,
"percent")
        frmSkidderResults.fgSkidderResults.TextMatrix(2, j + 2) = frmSkidderProfile.txtDistance(j).text
        frmSkidderResults.fgSkidderResults.TextMatrix(3, j + 2) = frmSkidderProfile.txtCI(j).text
        Else 'METRIC  = true
        frmSkidderResults.fgSkidderResults.TextMatrix(1, j + 2) = Format(Slope(j) / 100, "percent")
        frmSkidderResults.fgSkidderResults.TextMatrix(2, j + 2) = Format(Distance(j) * 0.3048, "fixed")
        frmSkidderResults.fgSkidderResults.TextMatrix(3, j + 2) = Format(CI(j) * 6.89475729317, "fixed")
        End If
    End If
'--- skid resistance ---
    If Arch = 1 Then        'skidding with an arch
        H = Yp / 12 - 1     'distance from ground to hook point
        l1 = 1              'distance from butt to choker hook point
        l2 = 0.2 * l3       'average lenght of contact of logs and ground
        C8 = 0.4 * l3 * T8  'distance from butt to center of gravity
        x = l3 - l1 - l2: If x = 1 Then x = 1.1 'avoid /0 in 2nd line down (suspended stem length)
        q1 = (1 - (H / x) ^ 2) ^ 0.5  '
        q2 = Sin(T) * H * ((l3 - l2 - C8) / x - 1)
        q3 = q1 * (C8 - l1) * Cos(T)
        q4 = u * H + l2 / 2 + x * q1
        q = TurnWt * (q3 + q2) / q4   'normal force of tree length log on soil
        P1 = TurnWt * Cos(T) - q      'normal force of logs
        P2 = TurnWt * Sin(T) + u * q  'tangential force of logs
    Else                              'skidding without an arch
        N7 = 0.2                        'log weight transferred to vehicle decimal fraction
        C7 = 0.9 + 1.667 * Tan(T)       'coefficient of resistance to skidding
        P1 = N7 * TurnWt * Cos(T)       'normal force of logs
        P2 = (1 - N7) * C7 * TurnWt * Cos(T) + TurnWt * Sin(T) 'tangential force of logs
        q = (1 - N7) * TurnWt * Cos(T)  'normal force of tree lenght log on soil
    End If

skipfriction:
    '--- MECHANICAL EFFICIENCY=.875 ---
    h1 = (Efficiency / 100) * HP
    'h1 = 0.875 * HP     'hp without torque converter
    'If TQ = 1 Then
    'h1 = h1 * 0.9   '*.9 FOR TORQUE CONVERTER
    'End If

    N1 = (SkidderW * (L - Xc) * Cos(T) - SkidderW * Yc * Sin(T) - P1 * Xp - P2 * Yp) / L
    N2 = SkidderW * Cos(T) + P1 - N1    'normal weight on rear tires loaded
    N1 = N1 + BALLAST * 2               'normal weight on front tires loaded
    If N1 < 100 Then                    'normal weight on front tires loaded
        MsgBox "ERROR: INSTABILITY - FRONT TIRES" & vbCrLf & "(try to reduce load or reduce slope)" _
        & vbCrLf & "(this problem was encountered while calculating segment # " & j
        frmSkidderResults.Visible = False
        frmSkidderMenu.Visible = True
        Exit Sub
    End If

    If Xp >= 0 Then                     '
        If (N1 / 2) > WR1 Or (N2 / 2) > WR2 Then
            MsgBox "WARNING: RATED TIRE LOAD EXCEEDED (pull equations not valid)" & vbCrLf _
                & "(this problem was encountered while calculating segment # " & j
            frmSkidderResults.Visible = False
            frmSkidderMenu.Visible = True
            Exit Sub
        End If
    End If

    If Xp < 0 Then
        'PRINT "N1/2"; N1 / 2; "N2/4"; N2 / 4; "WR2"; WR2
        If N1 / 2 > WR1 Or N2 / 4 > WR2 Then   'dual load assumed  = N2/2
            MsgBox "WARNING: RATED TIRE LOAD EXCEEDED (pull equations not valid)"
            frmSkidderResults.Visible = False
            frmSkidderMenu.Visible = True
            Exit Sub
        End If
    End If

    If u = 99 Then  ' full suspension reporting
        frmSkidderResults.fgSkidderResults.TextMatrix(6, j + 2) = "full suspension"
        frmSkidderResults.fgSkidderResults.TextMatrix(7, j + 2) = "full suspension"
        frmSkidderResults.fgSkidderResults.TextMatrix(8, j + 2) = "full suspension"
    Else
        If METRIC = False Then
            frmSkidderResults.fgSkidderResults.TextMatrix(6, j + 2) = Format(q, "fixed")
            frmSkidderResults.fgSkidderResults.TextMatrix(7, j + 2) = Format(P2, "fixed")
            frmSkidderResults.fgSkidderResults.TextMatrix(8, j + 2) = Format(Sqr((1.1 * P2) ^ 2 + P1 ^ 2), "fixed")
        Else 'metric = true
            frmSkidderResults.fgSkidderResults.TextMatrix(6, j + 2) = Format(q * (4.44822161526 / 1000), "fixed")
            frmSkidderResults.fgSkidderResults.TextMatrix(7, j + 2) = Format(P2 * (4.44822161526 / 1000), "fixed")
            frmSkidderResults.fgSkidderResults.TextMatrix(8, j + 2) = Format(Sqr((1.1 * P2) ^ 2 + P1 ^ 2) *
(4.44822161526 / 1000), "fixed")
        End If
    End If

    If Xp >= 0 Then
        C2 = C1 * B1 * D1 / (N1 * 0.5)      'Freitag numeric front tire loaded
        C3 = C1 * B1 * D1 / (N2 * 0.5)      'Freitag numeric rear tire loaded
        R5 = -0.1 * (N1 * 0.5 / WR1) + 0.22 / C2 + 0.2 'WHEEL RESISTANCE - FRONT
        R6 = -0.1 * (N2 * 0.5 / WR2) + 0.22 / C3 + 0.2 'WHEEL RESISTANCE - REAR
    End If

    If Xp < 0 Then
        C2 = C1 * B1 * D1 / (N1 * 0.5)      'Freitag numeric front tire loaded
        C3 = C1 * 2 * B1 * D1 / (N2 * 0.5)  'Freitag numeric rear tire loaded
        R5 = -0.1 * (N1 * 0.5 / WR1) + 0.22 / C2 + 0.2       'WHEEL RESISTANCE - FRONT
        R6 = -0.1 * (N2 * 0.5 / (2 * WR2)) + 0.22 / C3 + 0.2 'WHEEL RESISTANCE - REAR
    End If

    S = 0.09000001             ' slip loaded
    F = 0                      ' insure at least 1 time thru loop
    inc = 0.01                 ' slip increment
    FirstTime = True           ' conditional to refine slip one time


    While (F <= 0) Or (FirstTime)  'F = net breakout force available loaded
        If F > 0 Then
           S = S - inc
           inc = inc / 10
           FirstTime = False
        End If

        S = S + inc

        If S > 0.55 Then
            '--- CALCULATE AMOUNT CAPACITY IS EXCEEDED BY (C4)---
            C4 = (1.1 * P2 + SkidderW * Sin(T) - PP) / (1.1 * P2 + SkidderW * Sin(T)) * 100
            msg = "ERROR: LOAD EXCEEDS SKIDDER CAPACITY BY  " & Format(C4, "Fixed") & " %" & vbCrLf
            msg = msg & "ON PROFILE SEGMENT # " & j & vbCrLf
            msg = msg & "(try reducing the load or the slope or else try upgrading skidder)" & vbCrLf
        frmSkidderResults.Visible = False
        frmSkidderMenu.Visible = True
            MsgBox msg
            Exit Sub
        End If

        If Xp >= 0 Then
```

```
       P5 = 0.47 * (1 - Exp(-(0.2 * C2 * S))) + 0.28 * (N1 * 0.5 / WR1) 'gross thrust from front tire loaded
       P6 = 0.47 * (1 - Exp(-(0.2 * C3 * S))) + 0.28 * (N2 * 0.5 / WR2) 'gross thrust from rear tire loaded
       PP = N1 * (P5 - R5) + N2 * (P6 - R6)   ' 2 rear tires 'tractive pull parallel to ground loaded
    End If

    If Xp < 0 Then
       P5 = 0.47 * (1 - Exp(-(0.2 * C2 * S))) + 0.28 * (N1 * 0.5 / WR1)
       P6 = 0.47 * (1 - Exp(-(0.2 * C3 * S))) + 0.28 * (N2 * 0.5 / (2 * WR2))
       PP = N1 * (P5 - R5) + N2 * (P6 - R6)  ' 2 rear tires 'tractive pull parallel to ground loaded
    End If
      '--- BREAKOUT FORCE = 110 PERCENT OF SKID RESISTANCE ---
      F = PP - 1.1 * P2 - SkidderW * Sin(T)
Wend

'--- CALCULATE SKIDDER VELOCITY LOADED ---
If Xp >= 0 Then
     SkidderW = SkidderW + 2 * BALLAST
     V2 = Workpace / 100 * (1 - S) * (h1 * 33 * 1000) / (N1 * R5 + N2 * R6 + P2 + SkidderW * Sin(T))
     N5 = (SkidderW * (L - Yc) * Cos(T) - SkidderW * Xc * Sin(-T)) / L
     SkidderW = SkidderW - 2 * BALLAST
     N6 = SkidderW * Cos(T) - N5 'normal weight on rear tires unloaded
     N5 = N5 + 2 * BALLAST       'normal weight on front tires unloaded
     C5 = C1 * B1 * D1 / (N5 * 0.5) 'freitag numeric front tire unloaded
     C6 = C1 * B1 * D1 / (N6 * 0.5) 'freitag numeric rear tire unloaded
     R7 = -0.1 * (N5 * 0.5 / WR1) + 0.22 / C5 + 0.2 'rolling resitst front tire loaded
     R8 = -0.1 * (N6 * 0.5 / WR2) + 0.22 / C6 + 0.2 'rolling resitst rear tire loaded
End If

If Xp < 0 Then
     SkidderW = SkidderW + 2 * BALLAST
     V2 = Workpace / 100 * (1 - S) * (h1 * 33 * 1000) / (N1 * R5 + N2 * R6 + P2 + SkidderW * Sin(T))
     N5 = (SkidderW * (L - Yc) * Cos(T) - SkidderW * Xc * Sin(-T)) / L
     SkidderW = SkidderW - 2 * BALLAST
     N6 = SkidderW * Cos(T) - N5
     N5 = N5 + 2 * BALLAST
     C5 = C1 * B1 * D1 / (N5 * 0.5)
     C6 = C1 * 2 * B1 * D1 / (N6 * 0.5)
     R7 = -0.1 * (N5 * 0.5 / WR1) + 0.22 / C5 + 0.2
     R8 = -0.1 * (N6 * 0.5 / (2 * WR2)) + 0.22 / C6 + 0.2
End If

S1 = 0.09000001           'slip unloaded
F1 = 0                    'insure at least 1 time thru loop
inc = 0.01                'slip increment
FirstTime = True          'loop conditional

While (F1 <= 0) Or FirstTime
    If F1 > 0 Then
        S1 = S1 - inc
        inc = inc / 10
        FirstTime = False
    End If
      S1 = S1 + inc
    If S1 > 0.55 Then
        MsgBox "ERROR: ADVERSE SLOPE TOO STEEP TO CLIMB" & vbCrLf & "ON PROFILE SEGMENT # " & j _
        & vbCrLf & "(try to reduce slope angle)"
    frmSkidderResults.Visible = False
    frmSkidderMenu.Visible = True
        Exit Sub
    End If

    If Xp >= 0 Then
      P7 = 0.47 * (1 - Exp(-(2 * C5 * S1))) + 0.28 * (N5 * 0.5 / WR1)
      P8 = 0.47 * (1 - Exp(-(2 * C6 * S1))) + 0.28 * (N6 * 0.5 / WR2)
      P4 = N5 * (P7 - R7) + N6 * (P8 - R8)
      F1 = P4 - SkidderW * Sin(-T)
    End If

    If Xp < 0 Then
      P7 = 0.47 * (1 - Exp(-(2 * C5 * S1))) + 0.28 * (N5 * 0.5 / WR1)
      P8 = 0.47 * (1 - Exp(-(2 * C6 * S1))) + 0.28 * (N6 * 0.5 / (2 * WR2))
      P4 = N5 * (P7 - R7) + N6 * (P8 - R8)
      F1 = P4 - SkidderW * Sin(-T)
    End If
Wend


'--- CALCULATE SKIDDER VELOCITY UNLOADED ---
SkidderW = SkidderW + 2 * BALLAST

If Xp >= 0 Then
    V3 = Workpace / 100 * (1 - S1) * (h1 * 33 * 1000) / (N5 * R7 + N6 * R8 + SkidderW * Sin(-T))
End If

If Xp < 0 Then
    V3 = Workpace / 100 * (1 - S1) * (h1 * 33 * 1000) / (N5 * R7 + N6 * R8 + SkidderW * Sin(-T))
End If
```

```
    SkidderW = SkidderW - 2 * BALLAST

    '--- IF SKIDDER SPEEDS EXCEED MAX ALLOWABLE, DEFAULT TO MAX ALLOWABLE
    If V2 < 0 Or V2 > S3 Then V2 = S3
    If V3 < 0 Or V3 > S3 Then V3 = S3
    If V2 < V4 Or V3 < V4 Then
        MsgBox "ERROR: LOWEST GEAR HAS INSUFFICIENT POWER" & vbCrLf & "ON PROFILE SEGMENT # " & j
        frmSkidderResults.Visible = False
        frmSkidderMenu.Visible = True
        Exit Sub
    End If

    t2 = t2 + A2 / V2 + A2 / V3  'cumulative trip time
    V2 = V2 / 88
    V3 = V3 / 88
    S8 = D1 / (-30.94 + 6.62 * C3 - 0.13 * (0.45 * C3) ^ 2)
    S9 = D1 / (-30.94 + 6.62 * C6 - 0.13 * (0.45 * C6) ^ 2)

    If S8 < 0 Or S8 > 20 Then  'rear tire sinkage
        msg = "ERROR: EXCESSIVE SINKAGE " & vbCrLf & "ON PROFILE SEGMENT # " & j & vbCrLf
        msg = msg & "(try using wider tires, lighten the load, or wait for the soil conditions to change (dry season or
frozen soils))"
        MsgBox msg
        frmSkidderResults.Visible = False
        frmSkidderMenu.Visible = True
        Exit Sub
    End If

MeanContactStress  'calculate the mean contact stress (psi)

fgSkidderResults

Next j
    t2 = t2 + T3  'cumulative trip time and hook + unhook + deck time
    LBSPERHOUR = 60 * TurnWt / t2 * ((100 - Delay) / 100)        'LBS PER HOUR

    '=========================================================================
    '=  send results for total production summay to frmskidderResults label   =
    '=========================================================================
    frmSkidderResults.lblSkidderProduction(5).Caption = Format(TurnWt, "fixed") '* 0.4536; "kgs"
    frmSkidderResults.lblSkidderProduction(6).Caption = Format(t2, "fixed")
    frmSkidderResults.lblSkidderProduction(7).Caption = Format(LBSPERHOUR / LbsMbf, "fixed")
    frmSkidderResults.lblSkidderProduction(8).Caption = Format(LBSPERHOUR / LbsCunit, "fixed")
    frmSkidderResults.lblSkidderProduction(9).Caption = Format(F2 / LBSPERHOUR * LbsMbf, "Currency")
    frmSkidderResults.lblSkidderProduction(10).Caption = Format(F2 / LBSPERHOUR * LbsCunit, "Currency")

End Sub
Public Sub ConvertToMetric()
  ' convert to Metric Units  ********************************

    SkidderW = SkidderW * 0.4536    ' to kgs
    HP = HP * 0.7457   ' to Kw
    Xc = Xc * 2.54     ' to cm
    Yc = Yc * 2.54     ' to cm
    Xp = Xp * 2.54     ' to cm
    Yp = Yp * 2.54     ' to cm
    L = L * 2.54       ' to cm
    D1 = D1 * 2.54     ' to cm
    B1 = B1 * 2.54     ' to cm
    IP1 = IP1 * 6.895  ' to kPa
    IP2 = IP2 * 6.895  ' to kPa

    MinSpeed = MinSpeed * 1.609 ' to km/hr
    MaxSpeed = MaxSpeed * 1.609 ' to km/hr

    BALLAST = BALLAST * 0.4536  '  to kgs
    TurnWt = TurnWt * 0.4536    '  to kgs

    l3 = l3 * 0.3048            '  to m


End Sub
Public Sub ConvertToStandard()     ' make conversions back to English units  **********************

    SkidderW = SkidderW / 0.4536    ' back to lbs
    HP = HP / 0.7457   ' back to HP
    Xc = Xc / 2.54     ' back to in
    Yc = Yc / 2.54     ' back to in
    Xp = Xp / 2.54     ' back to in
    Yp = Yp / 2.54     ' back to in
    L = L / 2.54       ' back to in
    D1 = D1 / 2.54     ' back to in
    B1 = B1 / 2.54     ' back to in
    IP1 = IP1 / 6.895  ' back to psi
    IP2 = IP2 / 6.895  ' back to psi

    MinSpeed = MinSpeed / 1.609 ' back to mi/h
    MaxSpeed = MaxSpeed / 1.609 ' back to mi/h
    BALLAST = BALLAST / 0.4536  ' back to lbs

    l3 = l3 / 0.3048            ' back to feet
    TurnWt = TurnWt / 0.4536    ' back to lbs

End Sub
Private Sub skipfriction()
'--- MECHANICAL EFFICIENCY=.875 ---
    h1 = (Efficiency / 100) * HP
    'h1 = 0.875 * HP     'hp without torque converter
    'If TQ = 1 Then
    '    h1 = h1 * 0.9   '*.9 FOR TORQUE CONVERTER
    'End If
End Sub
Public Sub fgSkidderResults() 'populateFGSkidderResults()

    frmSkidderResults.Visible = True

    Dim entry As String

If TurnWt > 10 Then
    If METRIC = False Then
        frmSkidderResults.fgSkidderResults.TextMatrix(4, j + 2) = Format(N1, "fixed")
        frmSkidderResults.fgSkidderResults.TextMatrix(5, j + 2) = Format(N2, "fixed")
        frmSkidderResults.fgSkidderResults.TextMatrix(9, j + 2) = Format(V2, "fixed")
        frmSkidderResults.fgSkidderResults.TextMatrix(10, j + 2) = Format(V3, "fixed")
        frmSkidderResults.fgSkidderResults.TextMatrix(11, j + 2) = Format(S, "percent")
        frmSkidderResults.fgSkidderResults.TextMatrix(12, j + 2) = Format(S1, "percent")
        frmSkidderResults.fgSkidderResults.TextMatrix(13, j + 2) = Format(MCSF, "fixed")
        frmSkidderResults.fgSkidderResults.TextMatrix(14, j + 2) = Format(MCSR, "fixed")
        'frmSkidderResults.fgSkidderResults.TextMatrix(15, j + 2) = Format(2 * N2 / (D1 * B1), "fixed")
        'frmSkidderResults.fgSkidderResults.TextMatrix(16, j + 2) = Format(2 * N1 / (D1 * B1), "fixed")
    Else '  metric = true
        frmSkidderResults.fgSkidderResults.TextMatrix(4, j + 2) = Format(N1 * (4.44822161526 / 1000), "fixed")
        frmSkidderResults.fgSkidderResults.TextMatrix(5, j + 2) = Format(N2 * (4.44822161526 / 1000), "fixed")
        frmSkidderResults.fgSkidderResults.TextMatrix(9, j + 2) = Format(V2 * 1.609344, "fixed")
        frmSkidderResults.fgSkidderResults.TextMatrix(10, j + 2) = Format(V3 * 1.609344, "fixed")
        frmSkidderResults.fgSkidderResults.TextMatrix(11, j + 2) = Format(S, "percent")
        frmSkidderResults.fgSkidderResults.TextMatrix(12, j + 2) = Format(S1, "percent")
        frmSkidderResults.fgSkidderResults.TextMatrix(13, j + 2) = Format(MCSF * 6.89475728317, "fixed")
        frmSkidderResults.fgSkidderResults.TextMatrix(14, j + 2) = Format(MCSR * 6.89475728317, "fixed")
        'frmSkidderResults.fgSkidderResults.TextMatrix(15, j + 2) = Format(2 * N2 / (D1 * B1), "fixed")
        'frmSkidderResults.fgSkidderResults.TextMatrix(16, j + 2) = Format(2 * N1 / (D1 * B1), "fixed")
    End If
Else
End If

'save nominal loads for both loaded and unloaded turns
    If TurnWt > 10 Then
        NominalLoadFront(j) = N1
        NominalLoadRear(j) = N2
    Else
        NominalLoadFrontUnloaded(j) = N1
        NominalLoadRearUnloaded(j) = N2
    End If

If j = SEGMENTS Then
ClearResultInputData
Else
End If

End Sub
Public Sub MeanContactStress()
Dim TireType As Integer, TireRadius As Single
Dim ContactAreaN1 As Single, ContactAreaN2 As Single
On Error GoTo ErrorHandler

'estimating area of contact between tire and rigid surface Rule of thumb
Select Case PlyRating
    Case Is <= 6
        MCSF = 1.1 * IP1
        MCSR = 1.1 * IP2
    Case Is > 6, Is < 10
        MCSF = 1.15 * IP1
        MCSR = 1.15 * IP2
    Case Is >= 10, Is < 16
        MCSF = 1.2 * IP1
        MCSR = 1.2 * IP2
    Case Is >= 16
        MCSF = 1.25 * IP1
        MCSR = 1.25 * IP2
    Case Else
End Select

'estimating area of contact between tire and rigid surface Krick and Kroger
'If IP1 < 20 Then   'lower tire pressure equations
'    'calculate deflection
'    Select Case B1   ' Evaluate TireType.
'    Case Is < 22         ' TireType 1 size:18.4-34  inflation:15psi
'        DeflectionN1 = (0.000743 * (N1 / 2)) - (0.0000000193 * (N1 / 2) ^ 2)
'        DeflectionN2 = (0.000743 * (N2 / 2)) - (0.0000000193 * (N2 / 2) ^ 2)
'        TireRadius = D1 / 2
'        SectionHt = TireRadius - (RimDiameter / 2)
'    Case Is >= 22, Is < 27        ' TireType 3  size:24.5-32  inflation:15psi
'        DeflectionN1 = (0.000518 * (N1 / 2)) - (0.0000000146 * (N1 / 2) ^ 2)
'        DeflectionN2 = (0.000518 * (N2 / 2)) - (0.0000000146 * (N2 / 2) ^ 2)
'        TireRadius = D1 / 2
'        SectionHt = TireRadius - (RimDiameter / 2)
'    Case Is > 27          ' TireType 5  size:30.5l-32  inflation:15psi
'        DeflectionN1 = (0.000518 * (N1 / 2)) - (0.0000000138 * (N1 / 2) ^ 2)
'        DeflectionN2 = (0.000518 * (N2 / 2)) - (0.0000000138 * (N2 / 2) ^ 2)
'        TireRadius = D1 / 2
'        SectionHt = TireRadius - (RimDiameter / 2)
'    Case Else
'    End Select
'Else  'higher tire pressures equations
'    Select Case B1
'    Case Is < 22           ' TireType 2 size:18.4-34  inflation:25psi
'        DeflectionN1 = (0.000689 * (N1 / 2)) - (0.0000000268 * (N1 / 2) ^ 2)
'        DeflectionN2 = (0.000689 * (N2 / 2)) - (0.0000000268 * (N2 / 2) ^ 2)
'        TireRadius = D1 / 2
'        SectionHt = TireRadius - (RimDiameter / 2)
'    Case Is >= 22, Is < 27          ' TireType 4  size:24.5-32 inflation:25psi
'        DeflectionN1 = (0.000589 * (N1 / 2)) - (0.0000000266 * (N1 / 2) ^ 2)
'        DeflectionN2 = (0.000589 * (N2 / 2)) - (0.0000000266 * (N2 / 2) ^ 2)
'        TireRadius = D1 / 2
'        SectionHt = TireRadius - (RimDiameter / 2)
'    Case Is > 27             ' TireType 6  size:30.5l-32  inflation:25psi
'        DeflectionN1 = (0.000426 * (N1 / 2)) - (0.0000000112 * (N1 / 2) ^ 2)
'        DeflectionN2 = (0.000426 * (N2 / 2)) - (0.0000000112 * (N2 / 2) ^ 2)
'        TireRadius = D1 / 2
'        SectionHt = TireRadius - (RimDiameter / 2)
'    Case Else   ' Other values.
'    End Select
'End If
'Dim xcontactarean1
'ContactAreaN1 = 3.1415 * ((2 * TireRadius * DeflectionN1 - (DeflectionN1 ^ 2)) ^ 0.5) * ((2 * SectionHt / 2 *
DeflectionN1 - (DeflectionN1 ^ 2)) ^ 0.5)
''ContactArean1 = 2 * 3.1415 * ((TireRadius * (SectionHt / 2) * (DeflectionN1 ^ 2)) ^ 0.5)
'MCSF = (N1 / 2) / ContactAreaN1
'ContactAreaN2 = 3.1415 * ((2 * TireRadius * DeflectionN2 - (DeflectionN2 ^ 2)) ^ 0.5) * ((2 * SectionHt / 2 *
DeflectionN2 - (DeflectionN2 ^ 2)) ^ 0.5)
''ContactAreaN2 = 2 * 3.1415 * ((TireRadius * (SectionHt / 2) * (DeflectionN2 ^ 2)) ^ 0.5)
'MCSR = (N2 / 2) / ContactAreaN2


If METRIC = True Then
    MCSF = Format(MCSF * 6.89475729317, "fixed")   'convert to metric
    MCSR = Format(MCSR * 6.89475729317, "fixed")   'convert to metric
End If

Exit Sub

ErrorHandler:
MsgBox "error!"
frmMainMenu.Visible = True
End Sub

Public Sub ClearResultInputData()
    For j = 0 To SEGMENTS
        ResultInput(j) = Empty
    Next j
End Sub
```

FORWARDER MODULE

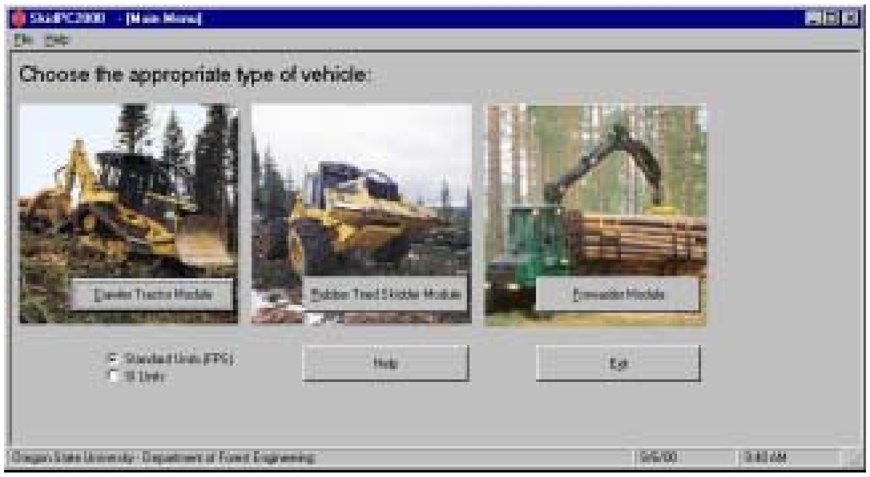
```
Option Explicit
Public Sub ForwarderAnalysis()
On Error GoTo ErrorHandler

    Xp = Xp * -1    'adjust XP value to place load point in front of rear wheel

    eror = 0
    If MinSpeed = 0 Then MinSpeed = 1.5
    If MaxSpeed = 0 Then MaxSpeed = 18
    If Workpace = 0 Then Workpace = 100

    If u = 0 Then u = 0.6

    V4 = MinSpeed * 88  'convert mi/h speed to fps
    S3 = MaxSpeed * 88  'convert mi/h speed to fps

    If Buttfirst = 1 Then T8 = 1 Else T8 = 1.5

    'h1 = 0.875 * HP: If TQ = 1 Then h1 = h1 * 0.9 '*.9 FOR TORQUE CONVERTER (axle horsepower)
    h1 = (Efficiency / 100) * HP ': If TQ = 1 Then h1 = h1 * (Efficiency / 100) '*.9 FOR TORQUE CONVERTER (axle
horsepower)

'--- compute rated tire loads ---
    WR1 = WR1
    WR2 = WR2
    'WR1 = 2.2046 * (0.75 * 0.00002735 * (6.894 * IP1) ^ 0.585 * (25.4 * B1) ^ 1.39 * (RimDiameter + B1) * 25.4)
    'WR2 = 2.2046 * (0.75 * 0.00002735 * (6.894 * IP2) ^ 0.585 * (25.4 * B1) ^ 1.39 * (RimDiameter + B1) * 25.4)


'--- ANALYSIS ---------------------------

'--- init variables ---
    i = SEGMENTS '# of segments
    t2 = 0: S = 0
'--- start analysis ---
For j = 0 To i - 1

    t1 = Slope(j)         ' must be percent slope
    A2 = Distance(j)      ' must be slope distance
    C1 = CI(j)            ' must be cone index
    T = Atn(t1 / 100)     ' slope beta

'--- output basic data
    If TurnWt > 10 Then
        If METRIC = False Then
        frmForwarderResults.fgForwarderResults.TextMatrix(1, j + 2) = Format(frmForwarderProfile.txtSlope(j).text /
100, "percent")
        frmForwarderResults.fgForwarderResults.TextMatrix(2, j + 2) = frmForwarderProfile.txtDistance(j).text
        frmForwarderResults.fgForwarderResults.TextMatrix(3, j + 2) = frmForwarderProfile.txtCI(j).text
        Else 'METRIC  = true
        frmForwarderResults.fgForwarderResults.TextMatrix(1, j + 2) = Format(Slope(j) / 100, "percent")
        frmForwarderResults.fgForwarderResults.TextMatrix(2, j + 2) = Format(Distance(j) * 0.3048, "fixed")
        frmForwarderResults.fgForwarderResults.TextMatrix(3, j + 2) = Format(CI(j) * 6.89475729317, "fixed")
        End If
    End If
'check for full suspension skidding
    If u = "FULL" Then
        P1 = TurnWt * Cos(T) 'normal force of logs
        P2 = TurnWt * Sin(T) 'tangential force of logs
        GoTo skipfriction    'go to subroutine to avoid log friction
    End If
'--- skid resistance ---
    Arch = 1                ' hardwire in arch use to simulate clambunk situation
    If Arch = 1 Then        'skidding with an arch
        H = Yp / 12 - 1     'distance from ground to hook point
        l1 = 1              'distance from butt to choker hook point
        l2 = 0.2 * l3       'average lenght of contact of logs and ground
        C8 = 0.4 * l3 * T8  'distance from butt to center of gravity
        x = l3 - l1 - l2: If x = 1 Then x = 1.1 'avoid /0 in 2nd line down (suspended stem length)
        q1 = (1 - (H / x) ^ 2) ^ 0.5
        q2 = Sin(T) * H * ((l3 - l2 - C8) / x - 1)
        q3 = q1 * (C8 - l1) * Cos(T)
        q4 = u * H + l2 / 2 + x * q1
        q = TurnWt * (q3 + q2) / q4   'normal force of tree length log on soil
        P1 = TurnWt * Cos(T) - q      'normal force of logs
        P2 = TurnWt * Sin(T) + u * q  'tangential force of logs
    Else                              'skidding without an arch
        End If

skipfriction:  'end goto point to skip skid resistances to simulate full suspension/forwarder situation

    '--- MECHANICAL EFFICIENCY=.875 ---
    h1 = (Efficiency / 100) * HP
    'h1 = 0.875 * HP     'hp without torque converter
    'If TQ = 1 Then
    '    h1 = h1 * 0.9   '*.9 FOR TORQUE CONVERTER
    'End If

    N1 = (ForwarderW * (L - Xc) * Cos(T) - ForwarderW * Yc * Sin(T) - P1 * Xp - P2 * Yp) / L
    N2 = ForwarderW * Cos(T) + P1 - N1             'normal weight on rear tires loaded
    N1 = N1 + BALLAST * 2                          'normal weight on front tires loaded

    If N1 < 100 Then                    'normal weight on front tires loaded
        MsgBox "ERROR: INSTABILITY - FRONT TIRES" & vbCrLf & "(try to reduce load or reduce slope)" _
        & vbCrLf & "this problem was encountered while calculating segment # " & j
        frmForwarderResults.Visible = False
        frmForwarderMenu.Visible = True
        Exit Sub
    End If

    If Xp >= 0 Then                     '
        If (N1 / 2) > WR1 Or (N2 / 4) > WR2 Then
            MsgBox "WARNING: RATED TIRE LOAD EXCEEDED (pull equations not valid)" & vbCrLf _
                & "this problem was encountered while calculating segment # " & j
        frmForwarderResults.Visible = False
        frmForwarderMenu.Visible = True
            Exit Sub
        End If
    End If

    If Xp < 0 Then
        If N1 / 2 > WR1 Or N2 / 4 > WR2 Then   'dual load assumed  = N2/2
            MsgBox "WARNING: RATED TIRE LOAD EXCEEDED (pull equations not valid)" & vbCrLf _
                & "this problem was encountered while calculating segment # " & j
        frmForwarderResults.Visible = False
        frmForwarderMenu.Visible = True
            Exit Sub
        End If
    End If

    If u = "FULL" Then  ' full suspension reporting
        frmForwarderResults.fgForwarderResults.TextMatrix(6, j + 2) = "full suspension"
        frmForwarderResults.fgForwarderResults.TextMatrix(7, j + 2) = "full suspension"
        'frmForwarderResults.fgForwarderResults.TextMatrix(8, j + 2) = "full suspension"
    Else
        If METRIC = False Then
            frmForwarderResults.fgForwarderResults.TextMatrix(6, j + 2) = Format(q, "fixed")
            frmForwarderResults.fgForwarderResults.TextMatrix(7, j + 2) = Format(P2, "fixed")
            'frmForwarderResults.fgForwarderResults.TextMatrix(8, j + 2) = Format(Sqr((1.1 * P2) ^ 2 + P1 ^ 2),
"fixed")
        Else 'metric = TRUE
            frmForwarderResults.fgForwarderResults.TextMatrix(6, j + 2) = Format(q * (4.44822161526 / 1000), "fixed")
'lbs to kN
            frmForwarderResults.fgForwarderResults.TextMatrix(7, j + 2) = Format(P2 * (4.44822161526 / 1000), "fixed")
'lbs to kN
            'frmForwarderResults.fgForwarderResults.TextMatrix(8, j + 2) = Format(Sqr((1.1 * P2) ^ 2 + P1 ^ 2) *
(4.44822161526 / 1000), "fixed") 'lbs to kN
        End If
    End If

    If Xp >= 0 Then
        C2 = C1 * B1 * D1 / (N1 * 0.5)      'Freitag numeric front tire loaded
        C3 = C1 * B1 * D1 / (N2 * 0.5)      'Freitag numeric rear tire loaded
        R5 = -0.1 * (N1 * 0.5 / WR1) + 0.22 / C2 + 0.2 'WHEEL RESISTANCE - FRONT
        R6 = -0.1 * (N2 * 0.5 / WR2) + 0.22 / C3 + 0.2 'WHEEL RESISTANCE - REAR
    End If

    If Xp < 0 Then
        C2 = C1 * B1 * D1 / (N1 * 0.5)      'Freitag numeric front tire loaded
        C3 = C1 * 2 * B1 * D1 / (N2 * 0.5)  'Freitag numeric rear tire loaded
        R5 = -0.1 * (N1 * 0.5 / WR1) + 0.22 / C2 + 0.2       'WHEEL RESISTANCE - FRONT
        R6 = -0.1 * (N2 * 0.5 / (2 * WR2)) + 0.22 / C3 + 0.2 'WHEEL RESISTANCE - REAR
    End If

    S = 0.09000001             ' slip loaded
    F = 0                      ' insure at least 1 time thru loop
    inc = 0.01                 ' slip increment
    FirstTime = True           ' conditional to refine slip one time


    While (F <= 0) Or (FirstTime)  'F = net breakout force available loaded
        If F > 0 Then
           S = S - inc
           inc = inc / 10
           FirstTime = False
        End If

        S = S + inc

        If S > 0.55 Then
            '--- CALCULATE AMOUNT CAPACITY IS EXCEEDED BY (C4)---
            C4 = (1.1 * P2 + ForwarderW * Sin(T) - PP) / (1.1 * P2 + ForwarderW * Sin(T)) * 100
            msg = "ERROR: LOAD EXCEEDS FORWARDER CAPACITY BY  " & Format(C4, "Fixed") & " %" & vbCrLf
            msg = msg & "ON PROFILE SEGMENT # " & j & vbCrLf
            msg = msg & "(try reducing the load or the slope or else try upgrading Forwarder)" & vbCrLf
            MsgBox msg
        frmForwarderResults.Visible = False
        frmForwarderMenu.Visible = True
            Exit Sub
        End If

        If Xp >= 0 Then
           P5 = 0.47 * (1 - Exp(-(0.2 * C2 * S))) + 0.28 * (N1 * 0.5 / WR1) 'gross thrust from front tire loaded
           P6 = 0.47 * (1 - Exp(-(0.2 * C3 * S))) + 0.28 * (N2 * 0.5 / WR2) 'gross thrust from rear tire loaded
           PP = N1 * (P5 - R5) + N2 * (P6 - R6)   ' 2 rear tires 'tractive pull parallel to ground loaded
        End If

        If Xp < 0 Then
           P5 = 0.47 * (1 - Exp(-(0.2 * C2 * S))) + 0.28 * (N1 * 0.5 / WR1)
           P6 = 0.47 * (1 - Exp(-(0.2 * C3 * S))) + 0.28 * (N2 * 0.5 / (2 * WR2))
           PP = N1 * (P5 - R5) + N2 * (P6 - R6)  ' 2 rear tires 'tractive pull parallel to ground loaded
        End If
          '--- BREAKOUT FORCE = 110 PERCENT OF SKID RESISTANCE ---
          F = PP - 1.1 * P2 - ForwarderW * Sin(T)
    Wend

    '--- CALCULATE Forwarder VELOCITY LOADED ---
    If Xp >= 0 Then
         ForwarderW = ForwarderW + 2 * BALLAST
         V2 = Workpace / 100 * (1 - S) * (h1 * 33 * 1000) / (N1 * R5 + N2 * R6 + P2 + ForwarderW * Sin(T))
         N5 = (ForwarderW * (L - Yc) * Cos(T) - ForwarderW * Xc * Sin(-T)) / L
         ForwarderW = ForwarderW - 2 * BALLAST
         N6 = ForwarderW * Cos(T) - N5                  'normal weight on rear tires unloaded
         N5 = N5 + 2 * BALLAST                          'normal weight on front tires unloaded
         C5 = C1 * B1 * D1 / (N5 * 0.5)                 'freitag numeric front tire unloaded
         C6 = C1 * B1 * D1 / (N6 * 0.5)                 'freitag numeric rear tire unloaded
         R7 = -0.1 * (N5 * 0.5 / WR1) + 0.22 / C5 + 0.2 'rolling resitst front tire loaded
         R8 = -0.1 * (N6 * 0.5 / WR2) + 0.22 / C6 + 0.2 'rolling resitst rear tire loaded
    End If

    If Xp < 0 Then
         ForwarderW = ForwarderW + 2 * BALLAST
         V2 = Workpace / 100 * (1 - S) * (h1 * 33 * 1000) / (N1 * R5 + N2 * R6 + P2 + ForwarderW * Sin(T))
         N5 = (ForwarderW * (L - Yc) * Cos(T) - ForwarderW * Xc * Sin(-T)) / L
         ForwarderW = ForwarderW - 2 * BALLAST
         N6 = ForwarderW * Cos(T) - N5
         N5 = N5 + 2 * BALLAST
         C5 = C1 * B1 * D1 / (N5 * 0.5)
         C6 = C1 * 2 * B1 * D1 / (N6 * 0.5)
         R7 = -0.1 * (N5 * 0.5 / WR1) + 0.22 / C5 + 0.2
         R8 = -0.1 * (N6 * 0.5 / (2 * WR2)) + 0.22 / C6 + 0.2
    End If

    S1 = 0.09000001           'slip unloaded
    F1 = 0                    'insure at least 1 time thru loop
    inc = 0.01                'slip increment
    FirstTime = True          'loop conditional

    While (F1 <= 0) Or FirstTime
        If F1 > 0 Then
            S1 = S1 - inc
            inc = inc / 10
            FirstTime = False
        End If
          S1 = S1 + inc
        If S1 > 0.55 Then
            MsgBox "ERROR: ADVERSE SLOPE TOO STEEP TO CLIMB" & vbCrLf & "ON PROFILE SEGMENT # " & j _
            & vbCrLf & "(try to reduce slope angle)"
        frmForwarderResults.Visible = False
        frmForwarderMenu.Visible = True
            Exit Sub
        End If

        If Xp >= 0 Then
          P7 = 0.47 * (1 - Exp(-(2 * C5 * S1))) + 0.28 * (N5 * 0.5 / WR1)
          P8 = 0.47 * (1 - Exp(-(2 * C6 * S1))) + 0.28 * (N6 * 0.5 / WR2)
          P4 = N5 * (P7 - R7) + N6 * (P8 - R8)
          F1 = P4 - ForwarderW * Sin(-T)
        End If

        If Xp < 0 Then
          P7 = 0.47 * (1 - Exp(-(2 * C5 * S1))) + 0.28 * (N5 * 0.5 / WR1)
          P8 = 0.47 * (1 - Exp(-(2 * C6 * S1))) + 0.28 * (N6 * 0.5 / (2 * WR2))
          P4 = N5 * (P7 - R7) + N6 * (P8 - R8)
          F1 = P4 - ForwarderW * Sin(-T)
        End If
    Wend


    '--- CALCULATE Forwarder VELOCITY UNLOADED ---
    ForwarderW = ForwarderW + 2 * BALLAST
```

```
    If Xp >= 0 Then
        V3 = Workpace / 100 * (1 - S1) * (h1 * 33 * 1000) / (N5 * R7 + N6 * R8 + ForwarderW * Sin(-T))
    End If

    If Xp < 0 Then
        V3 = Workpace / 100 * (1 - S1) * (h1 * 33 * 1000) / (N5 * R7 + N6 * R8 + ForwarderW * Sin(-T))
    End If

    ForwarderW = ForwarderW - 2 * BALLAST

    '--- IF Forwarder SPEEDS EXCEED MAX ALLOWABLE, DEFAULT TO MAX ALLOWABLE
    If V2 < 0 Or V2 > S3 Then V2 = S3
    If V3 < 0 Or V3 > S3 Then V3 = S3
    If V2 < V4 Or V3 < V4 Then
        MsgBox "ERROR: LOWEST GEAR HAS INSUFFICIENT POWER" & vbCrLf & "ON PROFILE SEGMENT # " & j
        frmForwarderResults.Visible = False
        frmForwarderMenu.Visible = True
        Exit Sub
    End If

    t2 = t2 + A2 / V2 + A2 / V3  'cumulative trip time
    V2 = V2 / 88
    V3 = V3 / 88
    S8 = D1 / (-30.94 + 6.62 * C3 - 0.13 * (0.45 * C3) ^ 2)
    S9 = D1 / (-30.94 + 6.62 * C6 - 0.13 * (0.45 * C6) ^ 2)

    If S8 < 0 Or S8 > 20 Then  'rear tire sinkage
        msg = "ERROR: EXCESSIVE SINKAGE " & vbCrLf & "ON PROFILE SEGMENT # " & j & vbCrLf
        msg = msg & "(try using wider tires, lighten the load, or wait for the soil conditions to change (frozen soils
or dry season))"
        MsgBox msg
        frmForwarderResults.Visible = False
        frmForwarderMenu.Visible = True
        Exit Sub
    End If

MeanContactStress  'calculate the mean contact stress (psi)

fgForwarderResults  'send results to flexgrid

Next j
    t2 = t2 + T3  'cumulative trip time and hook + unhook + deck time
    LBSPERHOUR = 60 * TurnWt / t2 * ((100 - Delay) / 100)        'LBS PER HOUR

    '========================================================================
    '=  send results for total production summay to frmForwarderResults label   =
    '========================================================================
    frmForwarderResults.lblForwarderProduction(5).Caption = Format(TurnWt, "fixed") '* 0.4536; "kgs"
    frmForwarderResults.lblForwarderProduction(6).Caption = Format(t2, "fixed")
    frmForwarderResults.lblForwarderProduction(7).Caption = Format(LBSPERHOUR / LbsMbf, "fixed")
    frmForwarderResults.lblForwarderProduction(8).Caption = Format(LBSPERHOUR / LbsCunit, "fixed")
    frmForwarderResults.lblForwarderProduction(9).Caption = Format(F2 / LBSPERHOUR * LbsMbf, "Currency")
    frmForwarderResults.lblForwarderProduction(10).Caption = Format(F2 / LBSPERHOUR * LbsCunit, "Currency")
Exit Sub

ErrorHandler:
    GlobalErrorHandler
End Sub
Public Sub ConvertToMetric()
  ' convert to Metric Units  ********************************

    ForwarderW = ForwarderW * 0.4536    ' to kgs
    HP = HP * 0.7457   ' to Kw
    Xc = Xc * 2.54     ' to cm
    Yc = Yc * 2.54     ' to cm
    Xp = Xp * 2.54     ' to cm
    Yp = Yp * 2.54     ' to cm
    L = L * 2.54       ' to cm
    D1 = D1 * 2.54     ' to cm
    B1 = B1 * 2.54     ' to cm
    IP1 = IP1 * 6.895  ' to kPa
    IP2 = IP2 * 6.895  ' to kPa

    MinSpeed = MinSpeed * 1.609 ' to km/hr
    MaxSpeed = MaxSpeed * 1.609 ' to km/hr

    BALLAST = BALLAST * 0.4536  '  to kgs
    TurnWt = TurnWt * 0.4536    '  to kgs

    l3 = l3 * 0.3048            '  to m


End Sub
Public Sub ConvertToStandard()     ' make conversions back to English units  **********************

    ForwarderW = ForwarderW / 0.4536    ' back to lbs
    HP = HP / 0.7457   ' back to HP
    Xc = Xc / 2.54     ' back to in
    Yc = Yc / 2.54     ' back to in
    Xp = Xp / 2.54     ' back to in
    Yp = Yp / 2.54     ' back to in
    L = L / 2.54       ' back to in
    D1 = D1 / 2.54     ' back to in
    B1 = B1 / 2.54     ' back to in
    IP1 = IP1 / 6.895  ' back to psi
    IP2 = IP2 / 6.895  ' back to psi

    MinSpeed = MinSpeed / 1.609 ' back to mi/h
    MaxSpeed = MaxSpeed / 1.609 ' back to mi/h

    BALLAST = BALLAST / 0.4536  ' back to lbs

    l3 = l3 / 0.3048            ' back to feet
    TurnWt = TurnWt / 0.4536    ' back to lbs

End Sub
Public Sub fgForwarderResults() 'populatetxtForwarderResults()

    frmForwarderResults.Show

    Dim entry As String

If TurnWt > 10 Then  'report for loaded conditions.
    If METRIC = False Then
        frmForwarderResults.fgForwarderResults.TextMatrix(4, j + 2) = Format(N1, "fixed")
        frmForwarderResults.fgForwarderResults.TextMatrix(5, j + 2) = Format(N2, "fixed")
        frmForwarderResults.fgForwarderResults.TextMatrix(9, j + 2) = Format(V2, "fixed")
        frmForwarderResults.fgForwarderResults.TextMatrix(10, j + 2) = Format(V3, "fixed")
        frmForwarderResults.fgForwarderResults.TextMatrix(11, j + 2) = Format(S, "percent")
        frmForwarderResults.fgForwarderResults.TextMatrix(12, j + 2) = Format(S1, "percent")
        frmForwarderResults.fgForwarderResults.TextMatrix(13, j + 2) = Format(MCSF, "fixed")
        frmForwarderResults.fgForwarderResults.TextMatrix(14, j + 2) = Format(MCSR, "fixed")
    Else 'metric = true
        frmForwarderResults.fgForwarderResults.TextMatrix(4, j + 2) = Format(N1 * (4.44822161526 / 1000), "fixed") 'lbs
to kN
        frmForwarderResults.fgForwarderResults.TextMatrix(5, j + 2) = Format(N2 * (4.44822161526 / 1000), "fixed") 'lbs
to kN
        frmForwarderResults.fgForwarderResults.TextMatrix(9, j + 2) = Format(V2 * 1.609344, "fixed")  'mi/h to km/h
        frmForwarderResults.fgForwarderResults.TextMatrix(10, j + 2) = Format(V3 * 1.609344, "fixed") 'mi/h to km/h
        frmForwarderResults.fgForwarderResults.TextMatrix(11, j + 2) = Format(S, "percent")
        frmForwarderResults.fgForwarderResults.TextMatrix(12, j + 2) = Format(S1, "percent")
        frmForwarderResults.fgForwarderResults.TextMatrix(13, j + 2) = Format(MCSF * 6.89475728317, "fixed") 'psi to
kPa
        frmForwarderResults.fgForwarderResults.TextMatrix(14, j + 2) = Format(MCSR * 6.89475728317, "fixed") 'psi to
kPa
    End If
    'removed the min/max inflation pressure
    'frmForwarderResults.fgForwarderResults.TextMatrix(15, j + 2) = Format(2 * N2 / (D1 * B1), "fixed")
    'frmForwarderResults.fgForwarderResults.TextMatrix(16, j + 2) = Format(2 * N1 / (D1 * B1), "fixed")
Else
End If

'save nominal loads for both loaded and unloaded turns
    If TurnWt > 10 Then
        NominalLoadFront(j) = N1
        NominalLoadRear(j) = N2
    Else
        NominalLoadFrontUnloaded(j) = N1
        NominalLoadRearUnloaded(j) = N2
    End If

If j = SEGMENTS Then
ClearResultInputData
Else
End If

End Sub
Public Sub MeanContactStress()
Dim TireType As Integer, TireRadius As Single
Dim ContactAreaN1 As Single, ContactAreaN2 As Single
On Error GoTo ErrorHandler

'estimating area of contact between tire and rigid surface Rule of thumb
Select Case PlyRating
    Case Is <= 6
        MCSF = 1.1 * IP1
        MCSR = 1.1 * IP2
    Case Is > 6, Is < 10
        MCSF = 1.15 * IP1
        MCSR = 1.15 * IP2
    Case Is >= 10, Is < 16
        MCSF = 1.2 * IP1
        MCSR = 1.2 * IP2
    Case Is >= 16
        MCSF = 1.25 * IP1
        MCSR = 1.25 * IP2
    Case Else
End Select

If METRIC = True Then
    MCSF = Format(MCSF * 6.89475729317, "fixed")   'convert to metric
    MCSR = Format(MCSR * 6.89475729317, "fixed")   'convert to metric
End If

Exit Sub


'estimating area of contact between tire and rigid surface
'If IP1 < 20 Then   'lower tire pressure equations
'    'calculate deflection
'    Select Case B1   ' Evaluate TireType.
'    Case Is < 22         ' TireType 1 size:18.4-34  inflation:15psi
'        DeflectionN1 = (0.000743 * (N1 / 2)) - (0.0000000193 * (N1 / 2) ^ 2)
'        DeflectionN2 = (0.000743 * (N2 / 2)) - (0.0000000193 * (N2 / 2) ^ 2)
'        TireRadius = D1 / 2
'        SectionHt = TireRadius - (RimDiameter / 2)
'    Case Is >= 22, Is < 27        ' TireType 3  size:24.5-32  inflation:15psi
'        DeflectionN1 = (0.000518 * (N1 / 2)) - (0.0000000146 * (N1 / 2) ^ 2)
'        DeflectionN2 = (0.000518 * (N2 / 2)) - (0.0000000146 * (N2 / 2) ^ 2)
'        TireRadius = D1 / 2
'        SectionHt = TireRadius - (RimDiameter / 2)
'    Case Is > 27          ' TireType 5  size:30.5l-32  inflation:15psi
'        DeflectionN1 = (0.000518 * (N1 / 2)) - (0.0000000138 * (N1 / 2) ^ 2)
'        DeflectionN2 = (0.000518 * (N2 / 2)) - (0.0000000138 * (N2 / 2) ^ 2)
'        TireRadius = D1 / 2
'        SectionHt = TireRadius - (RimDiameter / 2)
'    Case Else
'    End Select
'Else  'higher tire pressures equations  --always used for forwarder/clambunk
'    Select Case B1
'    Case Is < 22           ' TireType 2 size:18.4-34  inflation:25psi
'        DeflectionN1 = (0.000689 * (N1 / 2)) - (0.0000000268 * (N1 / 2) ^ 2)
'        DeflectionN2 = (0.000689 * (N2 / 4)) - (0.0000000268 * (N2 / 4) ^ 2)
'        TireRadius = D1 / 2
'        SectionHt = TireRadius - (RimDiameter / 2)
'    Case Is >= 22, Is < 27          ' TireType 4  size:24.5-32 inflation:25psi
'        DeflectionN1 = (0.000589 * (N1 / 2)) - (0.0000000266 * (N1 / 2) ^ 2)
'        DeflectionN2 = (0.000589 * (N2 / 4)) - (0.0000000266 * (N2 / 4) ^ 2)
'        TireRadius = D1 / 2
'        SectionHt = TireRadius - (RimDiameter / 2)
'    Case Is > 27             ' TireType 6  size:30.5l-32  inflation:25psi
'        DeflectionN1 = (0.000426 * (N1 / 2)) - (0.0000000112 * (N1 / 2) ^ 2)
'        DeflectionN2 = (0.000426 * (N2 / 4)) - (0.0000000112 * (N2 / 4) ^ 2)
'        TireRadius = D1 / 2
'        SectionHt = TireRadius - (RimDiameter / 2)
'    Case Else   ' Other values.
'    End Select
'End If

'ContactAreaN1 = 2 * 3.1415 * ((TireRadius * (SectionHt / 2) * DeflectionN1 ^ 2) ^ 0.5)
'MCSF = (N1 / 2) / ContactAreaN1
'ContactAreaN2 = 2 * 3.1415 * ((TireRadius * (SectionHt / 2) * DeflectionN2 ^ 2) ^ 0.5)
'MCSR = (N2 / 4) / ContactAreaN2

'If METRIC = True Then
'    MCSF = Format(MCSF * 6.89475729317, "fixed")   'convert to metric
'    MCSR = Format(MCSR * 6.89475729317, "fixed")   'convert to metric
'End If

'Exit Sub

ErrorHandler:
MsgBox "error!"
frmMainMenu.Visible = True
End Sub

Public Sub ClearResultInputData()
    For j = 0 To SEGMENTS
        ResultInput(j) = Empty
    Next j
End Sub
```

RUT DEPTH

```
Option Explicit

Public Sub UnloadedSkidderRutDepth()
'clear out any load on the skidder -the following values simulate an empty load
    TurnWt = 0.2
    l3 = 10
    u = 0.00002

'Run the Skidder Analysis again to get the unloaded specifications for the vehicle
    SkidderAnalysis
End Sub
Public Sub UnloadedForwarderRutDepth()
'clear out any load on the forwarder-the following values simulate an empty load
    TurnWt = 0.2
    l3 = 10
    u = "FULL"

'Run the Forwarder Analysis again to get the unloaded specifications for the vehicle
    ForwarderAnalysis

End Sub

Public Sub UnloadedCrawlerRutDepth()
'clear out any load on the crawler tractor -- the following values simulate an empty load
    TurnWt = 0.2
    l3 = 5
    u = 0.00002

'Run the Skidder Analysis again to get the unloaded specifications for the vehicle
    CrawlerAnalysis

End Sub
Public Sub CalculateEGP()
'from Wronski and Humphreys, jan 94 Journal of Forest Engineering
'calculate EGP of crawler tractor
TrackPitch = AShoe / TrackWd
TrackWheelDiam = 12
NGP = DynamicWL        'nominal ground pressure
'Mean Maximum Pressure
MMP = (1.26 * CrawlerW) / (2 * NTrackWheels * TrackWd * (TrackPitch * (2 * TrackWheelDiam)) ^ (0.5))
'Effective Ground Pressure of tracks
EGPTrack = 0.8 * NGP * ((MMP / (2 * NGP)) ^ (1.23)) 'effective ground pressure for crawler


End Sub
Public Sub CrawlerRutDepth()
Dim WheelPass, check As Integer, checklong As Long, lastpass, BCI, maxZ, TireDeflection, TireSectHt As Long
Dim TrackPass, RutDepthLoaded, RutDepthUnloaded
'check to see if max rut depth is greater than min vehicle clearance

For j = 0 To SEGMENTS - 1
    If MaxRutDepth(j) >= ClrMin Then
        MsgBox "Imobilization!  Maximum Rut Depth is" & vbCrLf _
        & "Greater than Minimum Vehicle Clearance"
        Exit Sub
    Else
    End If
Next j

'Perform Rut Depth Analysis
For j = 0 To SEGMENTS - 1
    MaximumRutDepth = MaxRutDepth(j)  'end rut depth or maximum
    TrackPass = 0         'start with the first wheel passing over
    BCI = CI(j)           'get cone index from profile data
    RutDepth = 0          'start using no rut for this pass
    TotalRutDepth = 0     'start with zero total rut depth

    If Slash(j) = 0 Then
        SlashFactor = 1.08      'raised up from zero to match up with the equation at 0 slash
    Else
        If METRIC = True Then
            Slash(j) = Slash(j) / 0.2097484 'convert tonne/ha to tons/acre
        Else
        End If

        SlashFactor = 1 / (0.0075 * Slash(j) + 0.93)
    End If

    check = 0


    Do
        'check to see what wheel is passing over the soil
        If TrackPass = (check * 2 + 0) Then
            DynamicWL = frmCrawlerResults.fgCrawlerResults.TextMatrix(9, j + 2) 'unloaded
        ElseIf TrackPass = (check * 2 + 1) Then
            DynamicWL = frmCrawlerResults.fgCrawlerResults.TextMatrix(13, j + 2) 'loaded
        End If

        'calculate EGP of crawler tractor
        'from Wronski and Humphreys, jan 94 Journal of Forest Engineering
        'Track Width common on low ground pressure machine 20-24 inches
        TrackPitch = AShoe / TrackWd 'common range of 12 - 18 inches
        TrackWheelDiam = RWDia      'diameter of track rollers or road wheels

        If NTrackWheels = 0 Then        'Code to deal with flexible track machines
            NTrackWheels = NRoadWheels
            Else
        End If

        'nominal ground pressure crawler weight and load /2 tracks / track area
        NGP = (GCW / 2) / (TrackWd * LCL)

        'Mean Maximum Pressure from Wronski
        MMP = (1.26 * CrawlerW) / (2 * NTrackWheels * TrackWd * (TrackPitch * (TrackWheelDiam)) ^ (0.5))

        'Effective Ground Pressure of tracks
        EGPTrack = 0.8 * NGP * ((MMP / (2 * NGP)) ^ (1.23)) 'effective ground pressure for crawler
        MMPNGPRatio = MMP / NGP

        NGP = EGPTrack * SlashFactor
        RutDepth = 4.6 * (150 / 2.54) / ((BCI / NGP) ^ (13 / 5))
        TotalRutDepth = (TotalRutDepth ^ 2 + RutDepth ^ 2) ^ (0.5)

        RutDepth = 0   'reset rutdepth for next pass

        check = Int(TrackPass / 2)
        TrackPass = TrackPass + 1  'add 1 to wheelpass for next wheel
        MachinePass = check '* 2    'convert wheel passes to round trip machine passes

    Loop Until Abs(TotalRutDepth) >= MaximumRutDepth Or MachinePass >= 500 'run over soil until max rut depth

    If MachinePass >= 499 Then
        frmCrawlerRutDepth.txtNumPasses(j).text = "more than 500"
        Else
        frmCrawlerRutDepth.txtNumPasses(j).text = Format(MachinePass, "0")
    End If

    'frmCrawlerRutDepth.txtAfterCI(j).Text = Format(ACI, "fixed")
    If METRIC = False Then
    frmCrawlerRutDepth.txtEndDepth(j).text = Format(TotalRutDepth, "fixed")
    Else
    frmCrawlerRutDepth.txtEndDepth(j).text = Format(TotalRutDepth * 2.54, "fixed") 'convert in to cm
    End If
Next j

frmCrawlerRutDepth.Show

End Sub

Public Sub WronskiSkidderRutDepth()
Dim WheelPass, check As Integer, checklong As Long, lastpass, BCI, maxZ, TireDeflection, TireSectHt As Long
Dim TrackPass, RutDepthLoaded, RutDepthUnloaded

ClrMin = D1 / 2  'estimate of min clearance = 1/2 tire diameter

For j = 0 To SEGMENTS - 1
    If MaxRutDepth(j) >= ClrMin Then
        MsgBox "Immobilization!  Maximum Rut Depth is" & vbCrLf _
        & "Greater than Minimum Vehicle Clearance"
        Exit Sub
    Else
    End If
Next j

UnloadedSkidderRutDepth
i = SEGMENTS

For j = 0 To i - 1

    MaximumRutDepth = MaxRutDepth(j)  'end rut depth or maximum
    WheelPass = 0         'start with the first wheel passing over
    BCI = CI(j)           'get cone index from profile data
    RutDepth = 0          'start using no rut

    If Slash(j) = 0 Then
        SlashFactor = 1.08      'raised up from zero to match up with the equation at 0 slash
    Else
        If METRIC = True Then
            Slash(j) = Slash(j) / 0.2097484 'convert tonne/ha to tons/acre
        Else
        End If
        SlashFactor = 1 / (0.0075 * Slash(j) + 0.93)

    End If

    TotalRutDepth = 0
    TireDeflection = DeflectionN1 + DeflectionN2 / 2 '2    'average tire deflection (front/rear)
    TireSectHt = SectionHt
    check = 0

    Do
        'check to see what wheel is passing over the soil
        If WheelPass = (check * 4 + 0) Then
            DynamicWL = NominalLoadFrontUnloaded(j)
        ElseIf WheelPass = (check * 4 + 1) Then
            DynamicWL = NominalLoadRearUnloaded(j)
        ElseIf WheelPass = (check * 4 + 2) Then
            DynamicWL = NominalLoadFront(j)
        ElseIf WheelPass = (check * 4 + 3) Then
            DynamicWL = NominalLoadRear(j)
        End If

        NGP = ((DynamicWL) / (B1 * D1)) * SlashFactor 'nominal ground pressure (dynamicWL is for axle or 2 wheels)

        RutDepth = 4.61 * (D1) / ((BCI / NGP) ^ (13 / 5))
        TotalRutDepth = (TotalRutDepth ^ 2 + RutDepth ^ 2) ^ (0.5)

        RutDepth = 0   'reset rutdepth for next pass

        check = Int(WheelPass / 4)
        WheelPass = WheelPass + 1  'add 1 to wheelpass for next wheel
        MachinePass = check '* 2    'convert wheel passes to round trip machine passes

    Loop Until Abs(TotalRutDepth) >= MaximumRutDepth Or MachinePass >= 500 'run over soil until max rut depth

    If MachinePass >= 499 Then
        frmSkidderRutDepth.txtNumPasses(j).text = "more than 500"
        Else
        frmSkidderRutDepth.txtNumPasses(j).text = Format(MachinePass, "0")
    End If

    'frmCrawlerRutDepth.txtAfterCI(j).Text = Format(ACI, "fixed")
    frmSkidderRutDepth.txtEndDepth(j).text = Format(TotalRutDepth, "fixed")
Next j
1558            'goto loop for rutting deeper than min clearance
frmSkidderRutDepth.Show

End Sub


Public Sub WronskiForwarderRutDepth()
Dim WheelPass, check As Integer, checklong As Long, lastpass, BCI, maxZ, TireDeflection, TireSectHt As Long
Dim TrackPass, RutDepthLoaded, RutDepthUnloaded


ClrMin = D1 / 2  'estimate of min clearance = 1/2 tire diameter

For j = 0 To SEGMENTS - 1
    If MaxRutDepth(j) >= ClrMin Then
        MsgBox "Imobilization!  Maximum Rut Depth is" & vbCrLf _
        & "Greater than Minimum Vehicle Clearance"
        Exit Sub
    Else
    End If
Next j

'clear out any load on the forwarder
    TurnWt = 0.2
    l3 = 10
    u = "FULL"

'Run the Forwarder Analysis
    ForwarderAnalysis

i = SEGMENTS

For j = 0 To i - 1

    MaximumRutDepth = MaxRutDepth(j)  'end rut depth or maximum
    WheelPass = 0         'start with the first wheel passing over
    BCI = CI(j)           'get cone index from profile data
    RutDepth = 0          'start using no rut

    If Slash(j) = 0 Then
        SlashFactor = 1.08      'raised up from zero to match up with the equation at 0 slash
    Else
        If METRIC = False Then
            Slash(j) = Slash(j) / 0.2097484 'convert tonne/ha to tons/acre
        Else
        End If

        SlashFactor = 1 / (0.0075 * Slash(j) + 0.93)
    End If

    TotalRutDepth = 0

    TireDeflection = DeflectionN1 + DeflectionN2 / 2 '2    'average tire deflection (front/rear)
    TireSectHt = SectionHt '8


    check = 0

    Do
        'check to see what wheel is passing over the soil
        If WheelPass = (check * 4 + 0) Then
            DynamicWL = NominalLoadFrontUnloaded(j)
        ElseIf WheelPass = (check * 4 + 1) Then
            DynamicWL = NominalLoadRearUnloaded(j)
        ElseIf WheelPass = (check * 4 + 2) Then
            DynamicWL = NominalLoadRearUnloaded(j)
        ElseIf WheelPass = (check * 4 + 3) Then
            DynamicWL = NominalLoadFront(j)
        ElseIf WheelPass = (check * 4 + 4) Then
            DynamicWL = NominalLoadRear(j)
        ElseIf WheelPass = (check * 4 + 5) Then
            DynamicWL = NominalLoadRear(j)
        End If

        NGP = ((DynamicWL) / (B1 * D1)) * SlashFactor 'nominal ground pressure

        RutDepth = 4.61 * (D1) / ((BCI / NGP) ^ (13 / 5))
        TotalRutDepth = (TotalRutDepth ^ 2 + RutDepth ^ 2) ^ (0.5)

        RutDepth = 0   'reset rutdepth for next pass

        check = Int(WheelPass / 6)  'divide by six wheelpasses (3 unload 3 load)
        WheelPass = WheelPass + 1  'add 1 to wheelpass for next wheel
        MachinePass = check '* 2    'convert wheel passes to round trip machine passes

    Loop Until Abs(TotalRutDepth) >= MaximumRutDepth Or MachinePass >= 500 'run over soil until max rut depth

    If MachinePass >= 499 Then
        frmForwarderRutDepth.txtNumPasses(j).text = "more than 500"
        Else
        frmForwarderRutDepth.txtNumPasses(j).text = Format(MachinePass, "0")
    End If

    frmForwarderRutDepth.txtEndDepth(j).text = Format(TotalRutDepth, "fixed")
Next j
1558            'goto loop for rutting deeper than min clearance
frmForwarderRutDepth.Show

End Sub
```

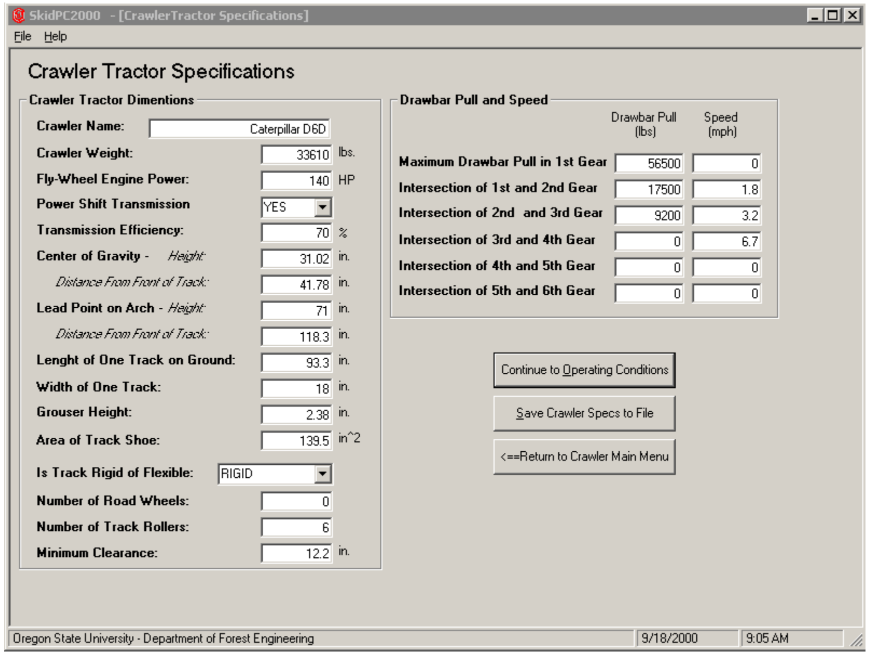
FORM SAMPLES



Choose the appropriate type of vehicle:

[✓ Standard Units (FPS)] SI Units

Oregon State University - Department of Forest Engineering | 9/6/00 | 9:40 AM


SkidPC2000 - [CrawlerTractor Specifications]
File Help
Continue to Operating Conditions
Save Crawler Specs to File
<==Return to Crawler Main Menu

Crawler Tractor Specifications

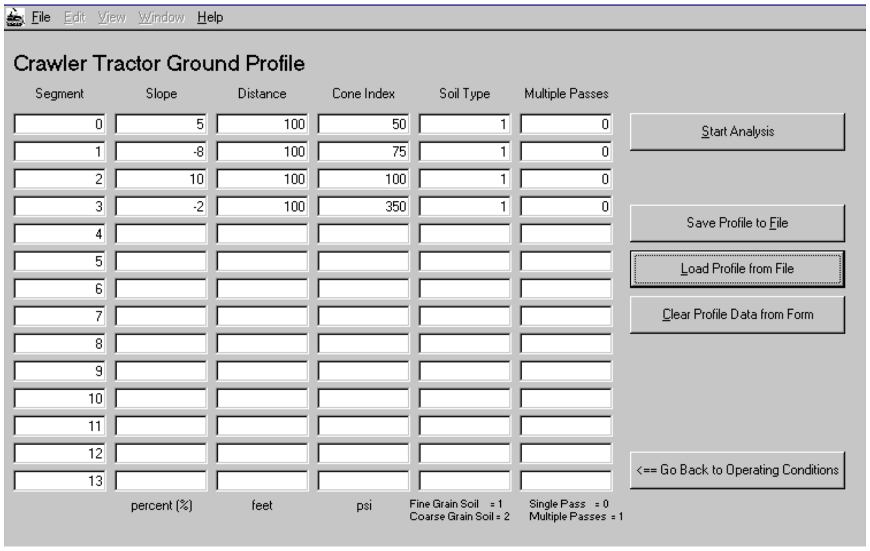
Crawler Tractor Dimentions

Crawler Name:	Caterpillar D6D	
Crawler Weight:	33610	lbs.
Fly-Wheel Engine Power:	140	HP
Power Shift Transmission	YES	
Transmission Efficiency:	70	%
Center of Gravity - *Height:*	31.02	in.
Distance From Front of Track:	41.78	in.
Lead Point on Arch - *Height:*	71	in.
Distance From Front of Track:	118.3	in.
Lenght of One Track on Ground:	93.3	in.
Width of One Track:	18	in.
Grouser Height:	2.38	in.
Area of Track Shoe:	139.5	in^2
Is Track Rigid of Flexible:	RIGID	
Number of Road Wheels:	0	
Number of Track Rollers:	6	
Minimum Clearance:	12.2	in.

Drawbar Pull and Speed

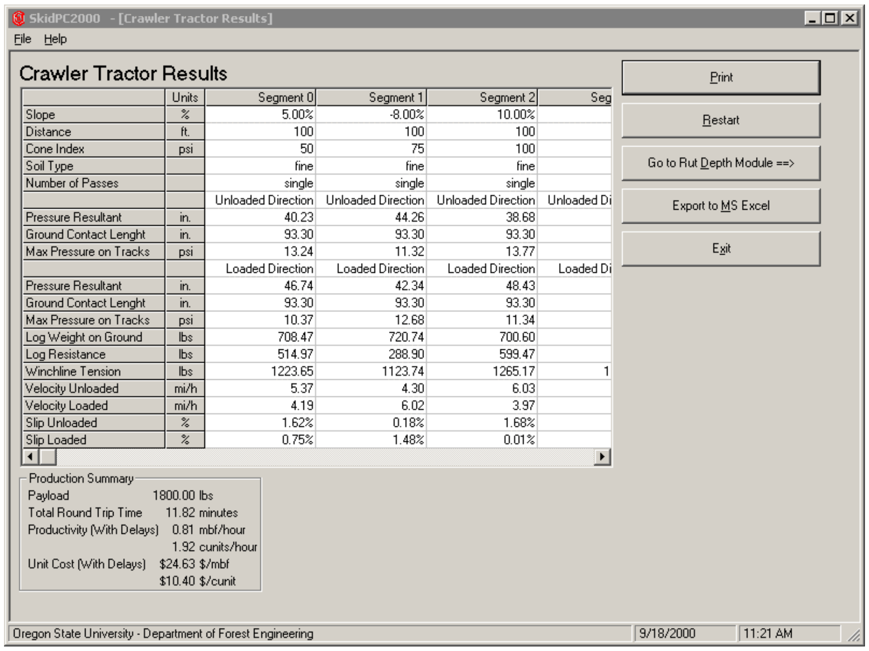
	Drawbar Pull (lbs)	Speed (mph)
Maximum Drawbar Pull in 1st Gear	56500	0
Intersection of 1st and 2nd Gear	17500	1.8
Intersection of 2nd and 3rd Gear	9200	3.2
Intersection of 3rd and 4th Gear	0	6.7
Intersection of 4th and 5th Gear	0	0
Intersection of 5th and 6th Gear	0	0

Oregon State University - Department of Forest Engineering | 9/18/2000 | 9:05 AM


File Edit View Window Help
Start Analysis
Save Profile to File
Load Profile from File
Clear Profile Data from Form
<== Go Back to Operating Conditions

Crawler Tractor Ground Profile

Segment	Slope	Distance	Cone Index	Soil Type	Multiple Passes
0	5	100	50	1	0
1	-8	100	75	1	0
2	10	100	100	1	0
3	-2	100	350	1	0
4					
5					
6					
7					
8					
9					
10					
11					
12					
13					
	percent (%)	feet	psi	Fine Grain Soil = 1 Coarse Grain Soil = 2	Single Pass = 0 Multiple Passes = 1


SkidPC2000 - [Crawler Tractor Results]
File Help
Print
Restart
Go to Rut Depth Module ==>
Export to MS Excel
Exit

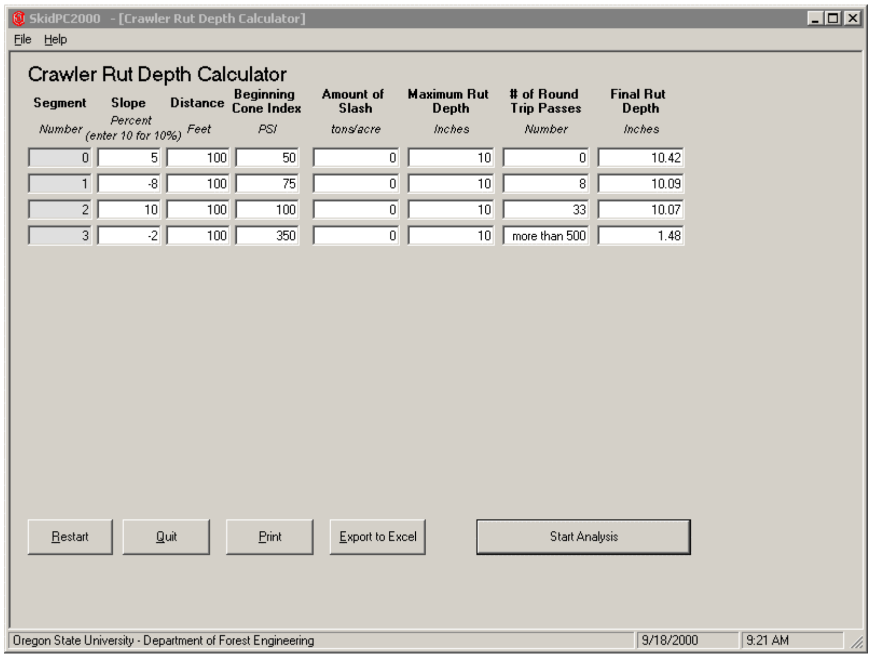
Crawler Tractor Results

	Units	Segment 0	Segment 1	Segment 2	Seg
Slope	%	5.00%	-8.00%	10.00%	
Distance	ft.	100	100	100	
Cone Index	psi	50	75	100	
Soil Type		fine	fine	fine	
Number of Passes		single	single	single	
		Unloaded Direction	Unloaded Direction	Unloaded Direction	Unloaded Di
Pressure Resultant	in.	40.23	44.26	38.68	
Ground Contact Lenght	in.	93.30	93.30	93.30	
Max Pressure on Tracks	psi	13.24	11.32	13.77	
		Loaded Direction	Loaded Direction	Loaded Direction	Loaded Di
Pressure Resultant	in.	46.74	42.34	48.43	
Ground Contact Lenght	in.	93.30	93.30	93.30	
Max Pressure on Tracks	psi	10.37	12.68	11.34	
Log Weight on Ground	lbs	708.47	720.74	700.60	
Log Resistance	lbs	514.97	288.90	599.47	
Winchline Tension	lbs	1223.65	1123.74	1265.17	1
Velocity Unloaded	mi/h	5.37	4.30	6.03	
Velocity Loaded	mi/h	4.19	6.02	3.97	
Slip Unloaded	%	1.62%	0.18%	1.68%	
Slip Loaded	%	0.75%	1.48%	0.01%	

Production Summary

Payload	1800.00 lbs
Total Round Trip Time	11.82 minutes
Productivity (With Delays)	0.81 mbf/hour
	1.92 cunits/hour
Unit Cost (With Delays)	$24.63 $/mbf
	$10.40 $/cunit

Oregon State University - Department of Forest Engineering | 9/18/2000 | 11:21 AM



Crawler Rut Depth Calculator

Segment Number	Slope Percent (enter 10 for 10%)	Distance Feet	Beginning Cone Index PSI	Amount of Slash tons/acre	Maximum Rut Depth Inches	# of Round Trip Passes Number	Final Rut Depth Inches
0	5	100	50	0	10	0	10.42
1	-8	100	75	0	10	8	10.09
2	10	100	100	0	10	33	10.07
3	-2	100	350	0	10	more than 500	1.48

Oregon State University - Department of Forest Engineering 9/18/2000 9:21 AM

APPRENDIX C: ONLINE HELP DOCUMENT