

**University of Alaska
Anchorage**

Jason Bailey
533 N Denali St
Palmer, AK 99645

Home 907 746 5117
Work 907 273 1715
baileyj@math.uaa.alaska.edu

Applied Software Development Project

Automating Traveling Cylinder Map Creation

Abstract

This report describes the methods used in creating a program to automate certain tasks involved in creating a traveling cylinder diagram. The program performs the mundane tasks involved in drawing the diagram, thus significantly reducing the number of random errors introduced. Hence, the engineer focuses on the task of drawing the tolerance lines.

The program consists of various components. These components are the graphical user interface, data model, report reader, drawing writer, and data processor. These components allow the engineer to specify depth ranges for the diagram, set the depth tic intervals, and set the intervals for the no-go circles. This modular design allows for future enhancements or changes without having to rewrite the whole program.

Also included are subsections which discuss the development of some of the more mathematically involved algorithms. These sections present an informal presentation of the logic upon which these algorithms are based.

Table of Contents

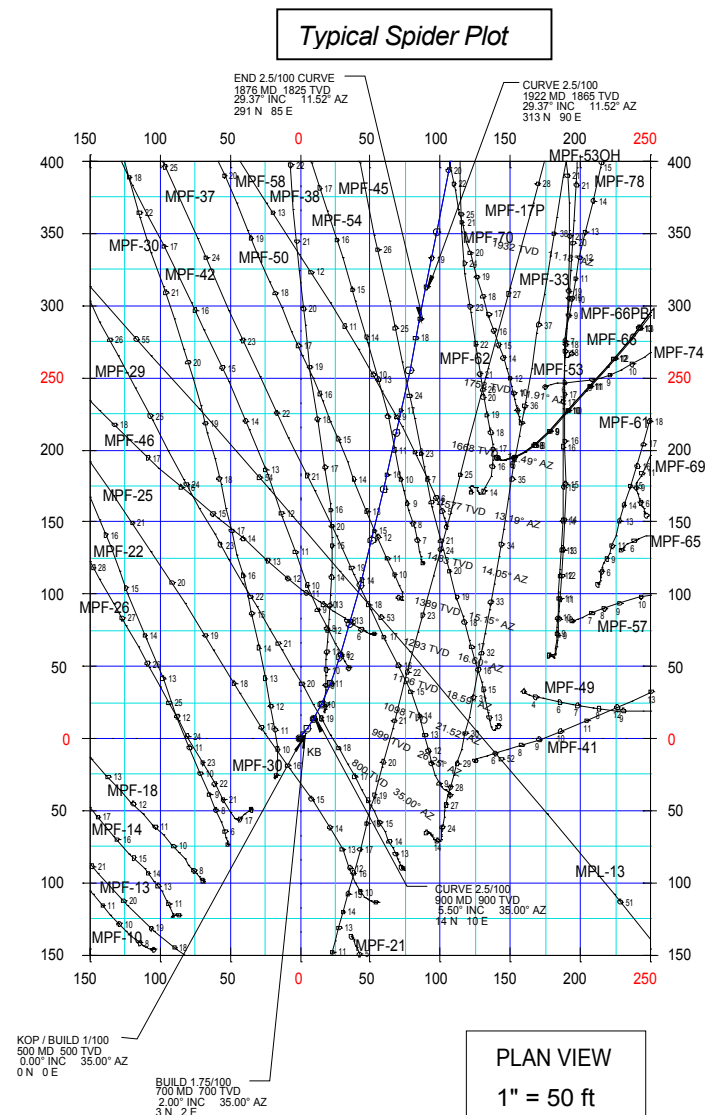
Abstract.....	1
Introduction	3
Review of Literature	5
Statement of Problem	6
Problem Solution	7
Analysis of the Solution	8
Graphical User Interface	8
Data Structure	9
Report Reader	10
Output Writer	11
Algorithm Development.....	11
Multiple Tangents.....	12
Tangents Between Circles.....	13
Limitations of Study	14
Conclusion.....	14
References	15
Bibliography	15

Introduction

Modern oil well drilling techniques involve a process known as directional drilling, which is a method of deviating the well from a vertical inclination and steering it towards the pools of oil within a reservoir. This technique has allowed production of many square miles of reservoir from a single drilling pad. Allowing oil companies to cut production costs and pipeline expenses while reducing their environmental footprint.

The drawback to these densely populated drilling pads stems from the possibility of drilling into an existing well. Should another well be hit there are a number of possible environmental, safety and economical concerns ranging from the loss of a productive well to a blowout with fatalities. For these reasons, it has become increasingly important that overlapping methods of collision avoidance be used.

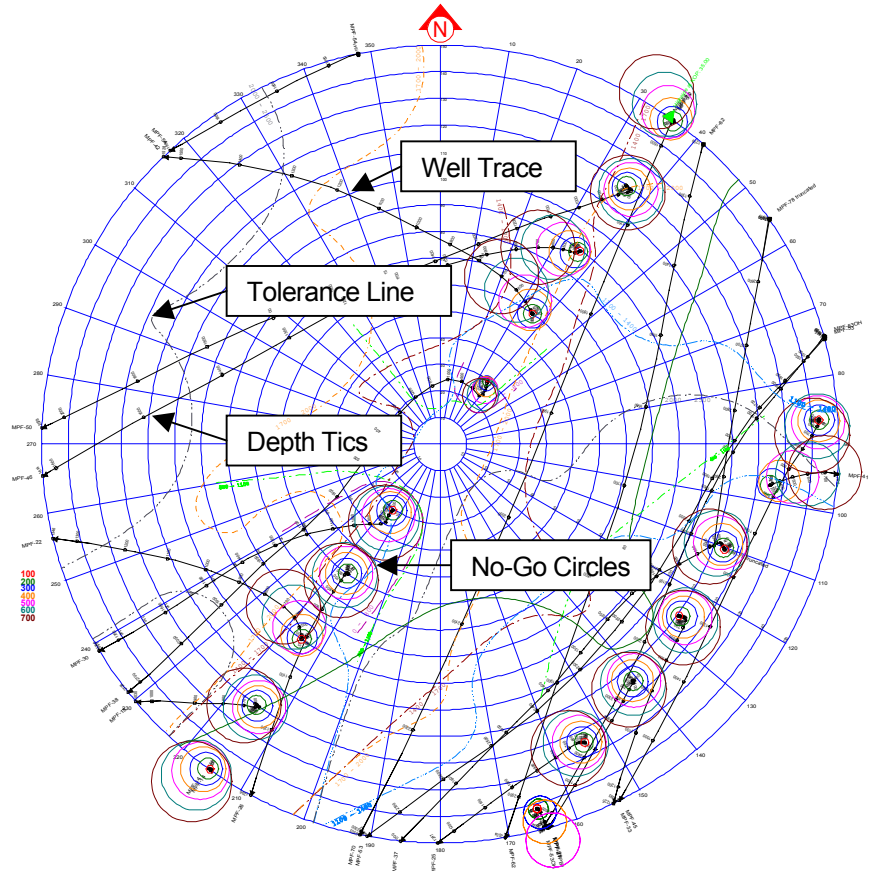
The first method is known as a spider plot, which shows the position of the wells on a 2D Cartesian coordinate plot with ticks on the well traces representing the depths. This view is the easiest for



those not well versed in anti-collision techniques to understand, but it can be deceptive, hence the need for additional method.

The traveling cylinder, the second method introduced, is a polar plot in which the planned well is always at the center of the diagram. The other well traces move about in relation to their distance and direction on the plane normal to the planned well. This method gives the directional driller a better idea of the position of other wells in relation to his location. Also, as he deviates from the plan he knows how far he can drift before a collision becomes a concern.

The greatest advantage to the traveling cylinder comes from the ability to add what we refer to as “no go” circles. These are circles, whose centers are place on the offset wells and whose radii are defined by adding the survey uncertainty of the existing well, the survey uncertainty of the planned well, and a safety factor. Once these circles are drawn, contour lines based on these “no-go” circles and arbitrary depth ranges can be added. The directional driller uses the contour lines, known as “Tolerance Lines,” in collision avoidance. When a directional driller gets too close to one of these lines, the oil company has to perform a risk assessment and plug-back the well if necessary.



Traveling Cylinder with no-go circles & tolerance

Review of Literature

I have found an extremely limited amount of information in print on the subject of traveling cylinders and tolerance lines. For this reason, much of my work is based on theory learned over my years as a well planning engineer. To cite exact reference for my knowledge proves quite difficult, whereas it is the culmination of countless conversations with drilling engineers from BP™, Arco™ and Schlumberger™.

Thurogood and Sawaryn, while explaining the theory of using traveling cylinder diagrams as tools for anti-collision, give a visualization aid to understand what the traveling cylinder represents.

To obtain a clearer understanding of what is happening, imagine the normal plane represented with a polystyrene disk set at right angles to the planned well. The planned well passes through the center of the disk. If the adjacent wells are represented as hot wires, then that burn traces in to the disk as the disk is pushed down the well. Wells that are nearly parallel to the planned well tend to have a single large hole burned in the disk; those with high convergence rates have lines that move rapidly across the disk with depth. (31)

The technique involved in constructing the no-go circles and tolerance lines became my next focus. I was able to find a document written by Hugh Williamson of BP Exploration's Drilling Technology Group. This document lays down some basic rules, but the example is so simplistic that much is left to interpretation. To illustrate his explanation, an extremely simple well was chosen. All offset wells depart rapidly and do not double back. It becomes quite obvious that this example is neither taken from a platform nor from a congested drill pad.

For validation of my mathematical formulae, I turned to Graphic Gems. Here I was able to locate a proof that was mathematically similar to the proof I had worked out on my own.

Statement of Problem

The process of creating close approach diagrams manually is very long and tedious; as a result, it is prone to many errors. It involves using engineering scales, a calculator, and a compass to create circles on a traveling cylinder diagram. There are scores of circles that must be drawn by hand for every well that appears on the diagram. More error is introduced when the circles' radii become too large for a standard compass and the engineer must resort to using string and a thumbtack to create the circles.

Typically it takes six to twelve hours to generate a traveling cylinder plot with tolerance lines. This is primarily due to the time spent calculating circle radii and drawing the circles. The quality control check of the diagram tends to take about an hour, as the reviewer, usually the engineer's supervisor, has to perform manual calculations of radii and check circles with a scale.

The calculation methods used in creating a normal plane traveling cylinder also create a problem. Since this method involves performing calculations at 100-foot intervals on the offset wells and back calculating where that point is on the subject well, the depth ranges for circles are not at the same increments. For example, *Well A* may have circles at a measured depth of 98, 175, 256, while *Well B* may have circles at a measured depth of 115, 228, 306. For this reason it would be of great benefit to normalize the data by interpolating to 100-foot increments.

Problem Solution

Since all the data needed to generate the traveling cylinder and the “no go” circles are available in an ASCII text “Anticollision Report” exported from Compass™, a program can be written that can parse out the pertinent data and load it into a linked list. The engineer will specify certain aspects such as depth ranges to analyze, interpolation intervals to use for the circles and depth tic marks. This data will then be used for calculating coordinates and radii of the “no go” circles as well as the coordinates of the well traces. After which, the program will produce a graphical representation of this data. In this case, I proposed that the output of the program would be a Data Exchange Format (DXF) file, as defined by Autodesk™, which can then be opened in a CAD program for the drawing of the actual tolerance lines. This program is a revision and extension of three earlier projects.

As this project is one that has evolved over the last five years with many enhancements along the way, my project report will showcase many of my earlier programs. These programs parsed reports from DEAP™, an engineering application that BP abandoned in favor of Compass™. These programs rendered images by using BasicCAD to create the images within the CAD environment. I decided to eliminate this step due to the poor performance of BasicCAD (the Y2K compliant version of the software takes three times longer to generate the drawing as did the previous version).

This project has been written in a variety of languages (i.e. Delphi 3/BasicCAD, Java/BasicCAD). The implementation, which I will be working on, is coded in Borland Delphi 5. This language is an object-oriented, visual Pascal. I have chosen this language to allow ease of integration into a larger program, designed to aid the well planning engineer, which has already been written in Delphi.

Analysis of the Solution

One of the earlier versions of this parsing program read the data file, parsed out the information needed and stored it in a temporary file. Then through several iterations of reading data from the a temporary file performing a few calculations and storing the results in another temporary file the desired output was collected and saved into the output file. This process worked, but it was clumsy, poorly documented, and difficult to modify. To resolve this problem, I have adopted the use of the linked list data structure. The primary advantage comes from the elimination of constant I/O activity. A secondary advantage involves code that is less cluttered and thus easier to read and maintain.

I've broken the process into a few general modules:

- Graphical User Interface (GUI)
- Data Structure
- Report Reader
- Output Writer

Graphical User Interface

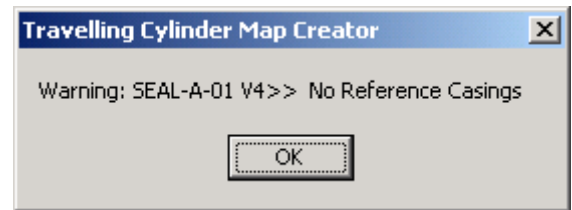
The graphical user interface has been designed in such away that all data can be easily input on one simple screen. Many of the default settings are actually loaded from data provided in other fields. For instance, once a “Traveling Cylinder File” has been selected, the “Drawing Exchange Format File” name is automatically input with a simple switch of the file extensions. In addition, the “Header Data” and “Survey

The screenshot shows the 'Travelling Cylinder Map Creator' application window. It features several sections: 'File Information' with fields for 'Travelling Cylinder File' and 'Drawing Exchange Format File', and a checkbox for 'Open DXF in DesignCAD'; 'Header Data' with fields for 'Well', 'Field', 'Rig', 'Client Engineer', 'Plan #', 'Scale', 'Map #', 'From Depth', and 'To Depth'; 'Survey Program' with a large text area; 'Drawing Options' with dropdown menus for 'Depth Tics' and 'No Go Circles'; a 'Convert' button; a 'Currently Analysing:' field; and a 'Status: Idle' indicator at the bottom.

Program” data are filled in based on data parsed out of the traveling cylinder file. This data can then be corrected if it is incorrect or incomplete.

Drop down boxes for the “Drawing Options” make it simple to enforce standards while still enabling the engineer to make the changes that he deems necessary. It is simple, yet powerful.

A status bar and the “Currently Analyzing” box provide a means for determining the progress of the analysis. I have provided a variety of messages that will popup for various known “errors” that can crop up. These are not fatal errors; rather they serve as a flag for the engineer to verify that his data was in fact correct. An example of this would be a warning that there are no reference casings on a particular well. This would prevent the proximity analysis necessary to create the no-go circles for certain sections of that particular well. The diagram can still be created.



Data Structure

By separating the data abstraction from the data parser, I have expanded the flexibility of the system. Should the data file change format we will be able make a new data parser without having to change the graphical user interface or the output writer. If we decide that we want the output written into any other format (a T-graph or an XY file), all we have to do is write a new output writer that understands the data structure.

The first step to design the data structure was to determine what data was available from the report. Then I narrowed down the structure to the data that I knew was important for generating the graph. Once I had this data, I then analyzed what was related in a one-to-one relationship and what was related in one-to-many relationship. From these lists I was able to establish which items would be in the parent linked-list and what data would be in the child linked list.

The linked-list appealed to me over any other data structure for a number of reasons. Surveys have a varying number of stations, so they are not well suited to an array. Trees do not lend themselves to this situation since the data is already sequential. For these reasons the linked-list fit the data best out of all the data structures available.

The first linked-list holds the data pertinent to the offset wells. Here we store the name of

```
TCSurveyRpt = class(TObject)
// Holds important data from a station
public
  next,prev : TCSurveyRpt;
  SMD       : real; //subject well
  STVD      : real; //subject well
  oMD       : real; //object well
  oTVD      : real; //object well
  aziTN     : real; //True North Azimuth
  aziHS     : real; //Highside Azimuth
  ctr2ctr   : real; //distance to no-go center
  allowDev  : real; //distance to no-go edge
  constructor create;
  destructor destroy; override;
end;
TCWell = CLASS(TObject)
  Next : TCWell; // Link to next well
  Head : TCSurveyRpt; // Child/Data List
  Tail : TCSurveyRpt; // Child/Data List
  Size : integer; // # items in Child list
  Name : String; // of the the well
  constructor create;
  destructor destroy; override;
  function AddStation(Sender: TObject; SMD :
    real; STVD : real; oMD : real; oTVD :
    real; aziTN : real; aziHS : real; ctr2ctr
    : real; allowDev : real): boolean;
end;
```

the well, a pointer to the next well, and pointers to the first and last nodes in the child list. It also provides the implement to load data into the child list.

The second linked-list, or child list, contains the actual data points. Each node represents a row of data from the report while the pertinent columns are stored in the corresponding variable within the data structure.

Report Reader

Report reader is the module of this program which will read the data file and load the data into the data structure. The primary function is to parse the data out of the reports. It will also be responsible for catching certain indicators within the file and must remind the engineer of the potential problem. In the example listed above in the section on the GUI, it was the report reader that told the GUI to send the error message alerting the engineer that casing references were missing.

The strict adherence to format simplifies the writing of the parser. The program reads a line and test for certain flags. If a flag presents itself then the program launches into a subroutine to handle that part of the report.

In the future, I will likely add more report parses. Currently, my program only supports files from Compass, but I plan to add a module that will read PowerPlan reports as well. Currently there is no demand for such a module, so I will reserve this enhancement until such time as it is deemed necessary.

Output Writer

The output writer is the key part of the program. The output is really what concerns the engineer, who really is not concerned with the journey so much as the destination. I have planned two output writers in my program. The first is the circle file creator. Circle files are used by a macro which I wrote for DesignCad. These files have all the information needed by the macro for it to create the traveling cylinder diagram.

The second output writer will create the DXF (data exchange format) file. DXF is a portable file that can be read by most graphics packages and all CAD packages. The primary advantage to DXF is speed. The DesignCad macros were too sluggish in the new versions of the software. A secondary advantage is that DXF does not tie you down to one software vendor since it is so widely supported.

A major challenge surfaces with the DXF format. As I hunt for a description of the DXF scheme, I chase many red herrings, but am eluded on the actual standard. Now I must reverse engineer the files and test my theories as to what is the actual format. Due to time and money constraints, the DXF writer was removed.

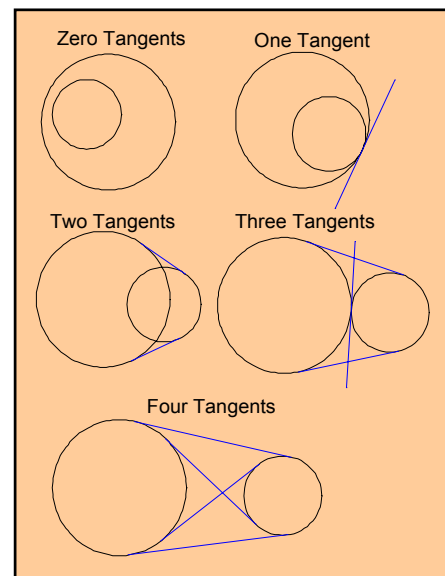
Algorithm Development

In the past, this program has just added circles to the traveling cylinder. I want to work towards an intelligent system that will not only draw the circles, but will

also draw the tolerance lines. Tackling something this ambitious slows down the turnaround time in software development. So, I've decided to just add one step, which will bring me closer to my goal. I have developed a method for adding tangent lines between the circles. Unfortunately, this is one of the many things I was unable to implement due to the cancellation of this project.

Multiple Tangents

Before I find the end points for the tangents, I need to find out if there are tangent lines. There are six possible scenarios for tangents between circles. If the circles have the same focus and radius, they are the same circle and therefore have an infinite number of tangents. The next we can look at the case where one circle encompasses the other; in this case there are no tangents. Another possibility is the circles may have exactly one tangent point if the smaller of the two circles is inside the larger and touches the larger circle in exactly one point. Two tangent lines will result when two circles overlap.



Three tangents result when the two circles touch at exactly one point, but the smaller is not inside the larger. Four tangents exist when the two circles do not intersect.

I then broke these up into two general cases based on whether it was necessary to add tangent lines. This was actually quite simple once I started to think about it. If the sum of the radius of the smaller circle and the distance between centers is greater than the radius of the larger circle, then I need to draw in the outer tangent lines. Otherwise there are no tangent that need be drawn.

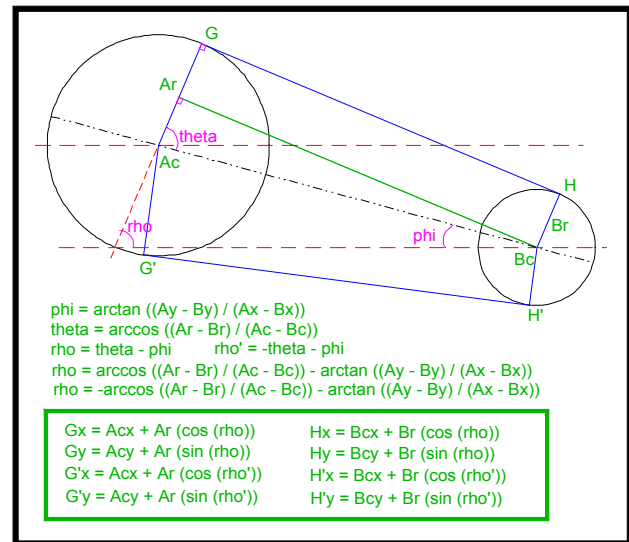
Tangents Between Circles

I searched online for resources that would help me write this algorithm, but I found that the solutions that were posted all involved very specialized cases and did not describe the actual trigonometry involved. I was resigned to solving the problem myself. I started up CAD and began to draw two circles

of different size with their center in different axis. I then began to draw in all the things that I knew about these circles. This consisted of the x and y coordinates and the radii. I then added the outer tangent line segments between the circles and labeled them G, G', H, and H'. Next I drew some lines of reference which consisted of a line through the centers of both circles, as well as two lines parallel to the X-axis through the centers of both circles.

I needed to find theta, the angle between the line connecting the centers of the circles and the point G; phi, the angle of rotation between the X-axis and the line connecting the centers; rho, the angle between the X-axis and the point G; and rho', the angle between the X-axis and the point G'.

I find theta to be the arccosine of the radius of circle A divided by the distance between the centers of the two circles. I calculate phi to simply be the slope of the line between the centers of the circles, so I simply take the arctangent of the change in y divided by the change in x. A quick examination of the drawing and a remembrance of the postulate that opposite interior angles of parallel lines are the equivalent, reveals that rho equals theta minus phi. Furthermore, rho prime must equal negative theta minus phi.



Limitations of Study

In the future, should someone decide to pursue this again, I will develop an algorithm that will be able to generate the tolerance lines. Linking arcs and tangents together will do this, but I must analyze the logic employed by the engineer. The difficult part will be converting this logic into something that can be represented in a programming language. I do believe that this will be something that should be done algorithmically and that there is no need to get heuristics involved.

Unfortunately, there is no funding for further development, so many of these ideas will likely never come to fruition. While working on this project I was told that I could no longer work on it, as it was no longer under the scope of my job. I had to quickly tie up the loose ends to provide my coworker with a program that he could use.

Conclusion

Through careful study of the problem and its current, manual solution, I have created an algorithmic approach to solve the problem. The solution appears quite complex at first; but after thoroughly examining the methodology it becomes quite clear that it is merely a chain of simple concepts and procedures which solve the problem.

The underlining purpose of computer science is algorithm development. Through the understanding of how to formulate efficient algorithms, computer scientists are able to solve engineering problems. I was able to solve the well planning

engineers' dilemma of random, human errors by providing a program that performed these tasks for the engineer. This program was a simple compilation of algorithms designed to perform the monotonous tasks.

The program was a success because it saves man-hours of time-consuming labor. It introduces a warning system that alerts the engineer to potential errors that he may otherwise overlook. Additionally, it has the potential for growth into a program that will create the entire diagram automatically, thus further reducing the time burden on the engineer.

References

Thorogood, J.L. and S.J. Sawaryn, "The Traveling-Cylinder Diagram: A Practical Tool For Collision Avoidance", SPEDE 19989 (Mar 1991): 31.

Glassner, Andrew S. Ed., Graphics Gems, Palo Alto: Academic Press, 1990.

Bibliography

Williamson, Hugh, "How to Draw an Anti-Collision Diagram", rev 2, XTP-Drilling Technology Group, BP Exploration, 3 Feb 1996.

Pacheco, Xavier and Steve Teixeira, Delphi 5 Developer's Guide, Indianapolis: Sams, 2000.

Kruse, Tondo, Leung, Data Structures & Program Design in C, 2nd ed. Englewood Cliffs: Prentice, 1997.

Version

3

SCHLUMBERGER OILFIELD SERVICES

Traveling Cylinder Map Creator

Users' Guide

TRAVELING CYLINDER MAP CREATOR

User's Guide

© Schlumberger Oilfield Services
3940 Arctic Blvd • Suite 300
Phone 907 273 1700 • Fax 907 561 8417

Table of Contents

Introduction

The traveling cylinder has become the industry standard for oil companies that are serious about collision avoidance while drilling an oil well. It has proved itself to be much more reliable than the simplistic spider plot. The strength of this polar plot lies in the functionality of adding no-go circles and their accompanying tolerance lines.

For the well planning engineer, there are few tasks more time consuming and monotonous as creating tolerance lines on a traveling cylinder document. This software takes great strides towards automating the more time intensive and tedious tasks. This allows the engineer to focus on the engineering rather than technician tasks.

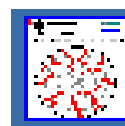
Close approach analysis has never been so easy. You will be able to provide close approach avoidance maps in minutes instead of hours.

Here is what the software does:

- Reads an ASCII close approach report from Compass™
- Creates a CAD compatible file (based on user criteria) that has
 - Well traces with depth ticks and the well's name
 - No go circles at specified intervals
 - Tangent lines between circles to further aid in creation of tolerance lines

Getting Started

No installation is necessary. You simply click on the TCyl.exe icon and the program will launch. You can download this file from Schlumberger Intranet at <http://www.anchorage.oilfield.slb.com/D-and-M/Tcyl.exe>.



The screenshot shows the 'Travelling Cylinder Map Creator' application window. It has a title bar with standard Windows controls. The main interface is divided into several sections:

- File Information:** Contains a 'Data Source' section with radio buttons for 'Compass' (selected) and 'PowerPlan'. Below this are two text input fields: 'Travelling Cylinder File' and 'DesignCAD ready CIR File'. There is also a checkbox for 'Create OpenDX file'.
- Header Data:** A table-like section with five columns: 'Well', 'Field', 'Rig', 'Client', and 'Plan #'. The 'Rig' field contains 'myRig' and the 'Client' field contains 'BP Exploration'. Below this is another row with fields for 'Scale' (20), 'Map #' (xxM-xx), 'From Depth', and 'To Depth'.
- Survey Program:** A large empty text area.
- Drawing Options:** A section with three dropdown menus: 'Ticks Blackout' (set to 500), 'No Go Circles' (set to No Inter), and 'Scale' (set to 20). A 'Convert' button is located to the right of these options.
- Status:** At the bottom left, it says 'Status: Idle'. At the bottom right, it says 'v0.9.2 Beta'.

You should see a window just as the one to the left of this text. This is where you specify the look of your traveling cylinder.

First one must select the traveling cylinder data file. Once this is selected, header fields and output file names are auto-populated.

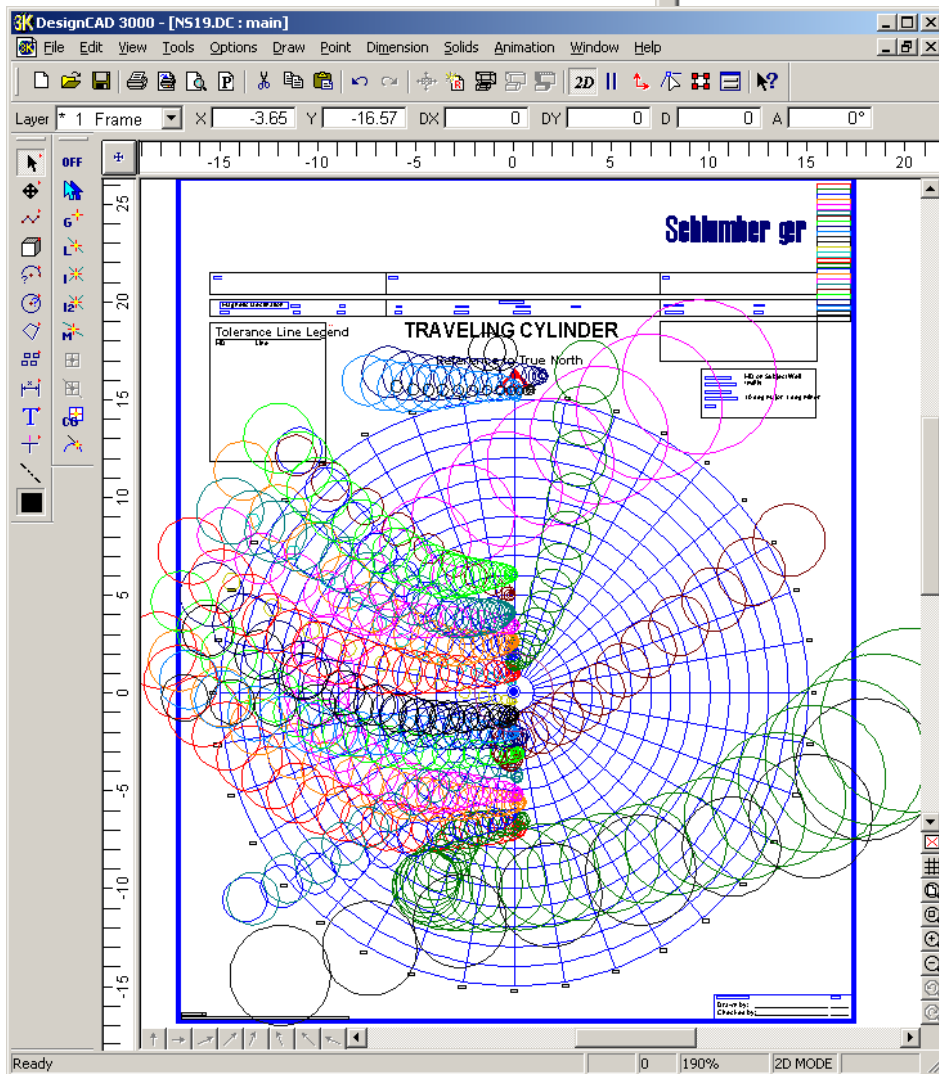
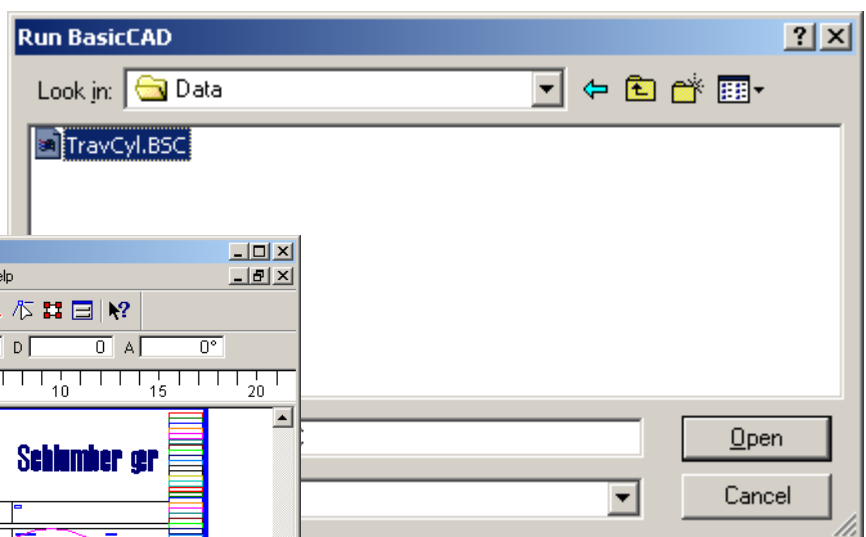
Next, fill in any data the application did not auto-populate or was incorrect.

Then, click “Convert” to create the CIR file.

Finally, in DesignCAD 3000, run the BasicCAD “TravCyl.BSC” program. The TravCyl.BSC program will ask you for the CIR file location and then it will draw a raw traveling cylinder file which must be edited by a trained Well Planning Engineer.

Interfacing with CAD

Run the TravCyl.BSC program by typing percent (%) and selecting the TravCyl.BSC file. Then type in the name of the file that you converted and the enter in the highside angle.



The program will then begin to draw the well paths, depth ticks, and circles. Once this is complete, the well planner can begin to edit the drawing by adding the tolerance lines.