Introduction

The challenge that is addressed by this paper came about in a very straightforward way. A technical colleague from the Irish Department of Transport asked the author how much clearance to the front and to the side would be required for an articulated truck to leave a testing lane and do a U-turn. This kind of information would typically be required where a new articulated-truck-testing lane was to be set up. A more general example of the same type of problem would involve checking whether a given articulated truck could negotiate a particular turn or roundabout. Articulated trucks come in a wide range of sizes and configurations, and it soon became apparent that even the identification of the worst-case scenario would require detailed consideration of a range of different cases. On the face of it, at least some aspects of the problem seemed suitable for solving by the use of a standard and ubiquitous spreadsheet application. The functionality of the spreadsheet that was developed is illustrated by a short video clip.

A Microsoft Excel spreadsheet workbook was developed that contained two worksheets: 'Dimensions' and 'Paths'. The former allowed the dimensions of the articulated truck to be specified and displayed to scale in plan view. The latter allowed steering information to be inputted, and the loci of principal points on the tractor and trailer to be plotted to scale on a diagram. A simple animation of the movement of the articulated truck was also included on the Paths sheet.

Dimensional inputs

The defining dimensions for an articulated truck are shown in Table 1. For the most part these are self-explanatory, although the term 'axle group' needs a little explanation.Tractors commonly have either one or two rear axles. Multiple axles can carry a greater load. If the number of rear axles is more than one, there will be some tyre dragging whenever a tractor turns. Because of this, some tractor units have the facility to raise one of the rear axles when the tractor is lightly loaded. In terms of turning behaviour, a group of rear axles can be represented by an equivalent single axle halfway between the first and the last of the active axles in a rear axle group. The turning characteristics of a tractor depend on the distance between the front steering axle and the centre of the rear axle group; for example, raising the frontmost of two tractor rear axles will lengthen the effective wheelbase for turning purposes. Semi-trailers can have a single rear axle or a rear axle group consisting of two, three or occasionally four rear axles. One or more of these could possibly be raised at light load to reduce tyre wear. As in the case of the tractor unit, for turning analysis purposes the rear axle group of a semi-trailer can be represented by an equivalent single rear axle that would be in the middle of the group of rear axles that are in use.

The 'maximum steering angle' is the maximum angle between the inside steering axle wheel and the centre-line of the tractor. The steering axle wheel on the inside of a turn always makes a greater angle with the centre line of the tractor than the steering axle wheel on the outside of the turn, as can be seen in Figure 1. The greater the maximum steering angle, the shorter will be the centre line turning radius of the tractor. Another significant dimensional parameter is the 'Steering axle width between contact centres'. This is the distance between the points where the king pin pivot axes of the steering axle intersect the road. It is approximately equal to the distance between the centres of the front tyre contact patches on the road. Table 2 lists truck dimensions that can be calculated from those listed in Table 1. These data were calculated automatically within the Dimensions sheet of the workbook by the use of the appropriate formulae.

The rear axle group of the tractor rotates about a turning centre as illustrated in Figure 1. The 'minimum centre-line turning radius' of the tractor is calculated from the tractor dimensions, including the maximum steering angle: it is the distance from the centre of the rear axle group to the current turning centre. The 'minimum cab front corner turning radius' is the distance from the outside front corner of the cab to the current turning centre.

The 'trailer angle' describes the angular relationship between the tractor and the semi-trailer: it is zero when the trailer is straight in line with the tractor. Figure 2 and Figure 3 illustrate different angular positions of the trailer. These diagrams were in fact XY scatter plots of data generated from the input dimensions and the trailer angle. The diagrams updated automatically when any of the input data changed. If any changes were made to the X- or Y-scales it would have been necessary to use the drag handles of this graph to ensure that the grid boxes

remained square. The main reference locations of the articulated truck were also identified in these figures. The origin of the articulated truck was taken to be the centre of the rear axle group of the tractor.

Steering control

With the aid of power steering it is possible to change the steering angle without any forward movement. It is more usual however (and much better for the tyres) to use the steering wheel to change the steering angle progressively as the truck moves. The analysis described here did not take dynamic effects into account and was based on finite displacements of the tractor's reference point (the centre of the tractor's rear axle group). For the purpose of defining and plotting the path travelled, a distance of 0.5 metres was a sufficiently small travel increment. However, still smaller sub-increments were used in the calculations to ensure high accuracy in the coordinates of each new plotted position along the path. In establishing the path of the articulated truck the steering lock was used (as a percentage value ranging from –100% to +100%). This corresponded to the amount by which a driver would have rotated the steering wheel. In the calculation sheet, provision was made for any arbitrary steering lock change at any path position, as well as for any arbitrary steering lock change from the previous position to the current position. Table 3 illustrates some of the path data from the Paths sheet. For the purpose of the example described here, the highest rate of steering-lock change with distance was taken as 20% per 0.5 metres, or 40% per metre of travel. Whenever the steering lock was zero, the tractor would travel in a straight line, and so the turning radius was infinite. To avoid displaying extremely large values for the turning radius, it was useful to use the inverse turning radius instead (zero when the tractor was travelling in a straight line). Keeping track of the position and direction of the tractor was a matter of cumulative addition and trigonometry.

Modelling of the paths traced out by the principal points on the trailer depended on being able to calculate how the position of the centre of the rear axle group varied as the towing king pin moved. The trailer was a trailing link attached to the king pin of the tractor and had two degrees of freedom: it could slide (or, more correctly, roll) in the direction of its centre-line, while also rotating about the centre of the rear axle group. Where the tractor travelled in a straight line, the centre of the rear axle group would typically have followed a curve. Of course, the tractor sometimes travelled in a curve and this complicated the path of the trailing link still further. This type of trailing-link motion was best modelled by considering a succession of very small increments of the overall movement, and this was the approach that was used.

For the example that is shown in Figure 4 an articulated truck was to leave a test lane and do a U-turn while taking the minimum space to the front and side of the test lane exit. At the start of the manoeuvre the articulated truck was in line with the testing lane and the tractor had advanced straight ahead until the centre of the rear axle group was level with the exit from the lane. As an immediate right turn would quickly bring the trailer body into contact with the right gate, the tractor first turned toward the left and then quickly changed to a full lock to the right. The tractor continued at full lock for some distance until its direction came close to the desired exit direction. The lock was then quickly reduced to zero so that the exit was in exactly the desired direction. A human driver would achieve this readily from his or her experience of truck driving. In the spreadsheet, the maximum steering change rate of 40% lock per metre was used, and the fine adjustment of the exit angle was achieved by starting and finishing with a lower absolute steering change rate. By some trial and error the desired exit direction of the tractor was achieved. Table 4 details the final steering adjustments from the Paths sheet of the workbook. The path distance between the successive positions was 0.5 metres. Just before the final adjustment the steering lock was –100%. This was reduced progressively to 0% in such a way that the truck left with the desired direction of 270º. The steering changes were made as the truck travelled, and therefore the values in the row entitled 'Steering lock increment at this point' remained at zero.

VBA Code Module

Simulating the turning of an articulated truck was a matter of plotting the loci of the paths of various points on the tractor and trailer in the two-dimensional space of the ground. The spreadsheet application came with standard trigonometric functions such as SIN or ATAN. It did not come with built-in functions for rotating one point about another, or for the more particular needs of modelling the turning of an articulated truck, for example calculating the path of a point that 'trailed' another. For efficiency in performing the calculations and to facilitate checking, it was necessary to write some functions in Visual Basic for Applications (VBA) code.

Describing the position of a point, such as a reference point on a truck, requires the specification of a coordinate pair. However, functions that could be used in the spreadsheet application only returned single-value results, and not coordinate-pair results. A simple workaround to address this issue was to write a subroutine that could give a required coordinate-pair result and then call this separately from two functions, one for the X-ordinate result and the other for the Y-ordinate result. In this way coordinate pair results could readily be written to the cells on the spreadsheet, using two cells for each result.

Such workarounds are commonly used in spreadsheet work. The wastefulness of carrying out the same calculations multiple times is counterbalanced by the convenience and the ease of use of the spreadsheet paradigm. In particular, once formulae and functions had been set up within a particular column corresponding to one position of an articulated truck, they could be rapidly copied to many columns, thereby automating the calculations for up to about 250 truck positions (the spreadsheet application allowed a maximum of 256 columns per worksheet).

A general subroutine that modelled a particular type of movement of an articulated truck might yield multiple result values, for example the final position of a reference point on the tractor (a coordinate pair), the orientation of the tractor (a single value) and the angle between the tractor and the trailer (a single value). Here again, the approach used was to define a single subroutine to carry out the calculations and call it from a separate function for each result value. For the example mentioned, there would therefore have been four separate functions yielding the final X-ordinate, Y-ordinate, tractor orientation and the angle between the tractor and the trailer.

The custom VBA procedures that were written for the truck-turning spreadsheet are described in Table 5 (the complete code listing is included in Appendix A). Some notes and comments relating to these are given below.

Rotate

This subroutine allows a point to be rotated about another point and makes use of trigonometric principles in a straightforward way. Separate functions RotatedX and RotatedY are used to call the subroutine for the X- and Y-ordinates.

Trailing

This subroutine has been written to calculate the new position of the tail end of a trailing link when the head is moved through a distance in a straight line. The underlying principle is that, for an infinitesimal displacement of the head in any direction, the new position of the tail will lie on a straight line that joins the original position of the tail to the new position of the head, as illustrated in Figure 5. For a finite displacement of the head in a straight line, the locus of the tail is generally a curve. Dividing the path of the head into a number of small subpaths and recalculating the position of the tail at the end of each can approximate this curve. For modelling the movement of an articulated truck, a subpath length of 0.01 metres was found to be adequately small and so, within the subroutine, the travel of the head was automatically divided into a number of subpaths that made the length of each less than or equal to this value. Five separate calling functions were written to allow the individual results from the subroutine to be inserted into cells of the spreadsheet: RotationByRadiusX, RotationByRadiusY, RotationByRadiusXCentre, RotationByRadiusYCentre and RotationByRadiusOutDir.

RotationByRadius

For the tractor of an articulated truck – or indeed for any non-articulated, front-axle-steered vehicle – there will be a specific turning radius associated with any particular position of the steering wheel (or amount of steering). As can be seen in Figure 1, the turning centre (or centre of rotation) lies on the centre-line of the rear axle if there is only one rear axle, or on a line that is parallel to the rear axles and passes through the centre of the rear axle group. The turning centre also lies, or should lie, on the axis of each of the steered front wheels. In a well-designed vehicle the outside steered wheel, which is at a greater radius from the turning centre, will always be at a smaller angle to the vehicle's centre-line than the inner steered wheel. If this ideal is not exactly achieved, tyre wear will be increased, but there will nonetheless be an equivalent effective turning radius for each amount of steering.
Subroutine RotationByRadius calculates the new position of a reference point that is rotated about a centre with a given radius and through a given path distance. It also calculates the

coordinates of the centre of rotation and the direction in which the path is pointing at the end of the travel. Separate calling functions – RotationByRadiusX, RotationByRadiusY, RotationByRadiusXCentre, RotationByRadiusYCentre and RotationByRadiusOutDir – are used to insert the outputs of the subroutine into cells on the spreadsheet. It should be noted that it is the inverse of the turning radius, rather than the turning radius directly, that is used in the subroutine.

Steer

Subroutine Steer builds on the functionality of subroutine RotationByRadius, which it calls, and allows two inverse turning radii to be specified: one at the beginning and the other at the end of the travel. Within the subroutine it is assumed that the inverse turning radius varies linearly with the distance travelled: this is a fair approximation if the steering wheel is being turned at a steady rate with respect to distance travelled (turning radius itself could not be considered to vary linearly with the distance travelled). Separate calling functions, SteerX, SteerY, SteerXCentre, SteerYCentre and SteerOutDir are used to insert the outputs of the subroutine into cells on the spreadsheet.

SteerAndTrail

Subroutine SteerAndTrail, like subroutine Steer, builds on the functionality of subroutine RotationByRadius, which it calls, and allows beginning and ending inverse turning radii to be specified. However, SteerAndTrail calculates the new position and direction of the trailer at the end of the movement by using the functionality of Trailing, which it also calls. This is a good example of how quite a powerful calculation can be built up from simpler components. Whereas Trailing strictly applies for straight-line motion, it is called within an iterative process where the incremental travel does not exceed 0.01 metre and the coordinates of each successive position of the tractor towing point are calculated precisely by RotationByRadius. Separate calling functions, TrailingX, TrailingY and TrailingDirDeg are used to insert the outputs of the subroutine into cells on the spreadsheet.

TimeTick

This tiny subroutine yields the use of the computer's processor to other processes for a period of 100 milliseconds. This provided the time delay between successive views of the articulated truck's position in the simple animation on the Paths sheet.

VBA code on the Paths sheet

In VBA code that was specific to the Paths sheet a Boolean (or logical) variable TimerOn was declared. The value of this variable (either True or False) determined whether the animation of the plan view of the truck on the sheet was on or off.

CommandButton1_MouseDown

When the Advance button on the Paths sheet was pressed, this event procedure set the variable TimerOn to True, it selected the cell on the Paths sheet that contained the position of the truck and proceeded to increment the position by one within an iterative loop that also included a delay of 100 milliseconds (produced by calling TimeTick) as long as TimerOn was True. Whenever the position exceeded 250 it was reset to 1.

CommandButton1_MouseUp

Whenever the Advance button was released, this event procedure set TimerOn to False, which has the effect of terminating the iteration loop within CommandButton1_MouseDown and thus stopped the animation.

Organisation of the workbook

The dimensions sheet

The first sheet of the workbook, Dimensions, contained the truck-defining dimensions and the derived dimensions, as in Table 1 and Table 2. The derived dimensions updated automatically whenever one of the defining dimensions was changed. For convenience, variable names were assigned to the cells containing the defining and dependent dimensions, for example, Len_AE for the value 'King pin to tractor rear axle group centre point'. Variable names like this could be referred to in a formula within a cell anywhere in the workbook.

The 'Maximum inverse centre-line turning radius' was the last of the dependent dimensions and was a key parameter in the steering calculations. This corresponded to 100% steering lock (positive for right turn and negative for left turn). It was calculated as the inverse of the 'Minimum centre-line turning radius,' which, in turn, was calculated using the following cell formula:
=Steering_Half_Width+Tractor_FAxle_to_Tractor_RAxle/TAN(Max_Steering_Angle*PI()/180)
where the symbols had the following meanings:

Steering_Half_Width	*Half the steering axle width between the tyre contact patch centres*
Tractor_Faxle_to_Tractor_RAxle	*The distance between the tractor front axle centre point and the tractor rear axle group centre point*
Max_Steering_Angle	*The maximum angle between the inside steering axle wheel and the centre-line of the tractor*
PI()/180	*This was the mathematical constant pi divided by 180. As a multiplier it converted the steering angle in degrees to radians.*

The Dimensions sheet also contained a region headed 'Default truck position (tractor heading straight in Y-direction with rear axle group centre point at Origin)'. Part of this is shown in Table 6. In this region, reference points on the tractor and trailer were defined. Cell notes described each point: for example, the cell note for point E read 'Centre point of tractor rear axle group'. The first data row provided the coordinates of 21 reference points on the tractor and trailer when the truck was in the default position. The second data row contained the coordinates of the same reference points where the tractor was still in the default position but where the trailer was rotated by a specified angle (the trailer angle) from its default position in line with the tractor. All the data values were automatically updated whenever one of the defining dimensions was changed or the trailer angle specified in the first cell of the row was changed.

Truck default position plan view

The Dimensions sheet also contained a plan view of the tractor and trailer, Figure 2, with the tractor in the default position and with a specifiable trailer angle. The data were taken from the region referred to in the paragraph above. For instance, the coordinates of the reference points HL, BL, BR, HR, HL, JL, JR and HR, taken in that order, defined the outline of the tractor and the back of the tractor cab. This group of coordinates was set out on the sheet under the heading 'Tractor' and the data were linked to the Default Truck Position table (see Table 6. The data of this group were the source data for the blue line that represented the tractor in the diagram, Figure 2 or Figure 3. Likewise individual points such as the towing king pin, A, or the trailer front right corner, FR, were plotted. The axes of the scatter plot graph needed to be set up appropriately to display all the data. Also, it was necessary to manually adjust the relative height and width of the graph area so that a unit in the X-direction had the same length as a unit in the Y-direction. Until this was done the tractor and trailer might have appeared distorted, with the right angles at the corners appearing as though they were not right angles.

The paths sheet

The second sheet of the workbook, Paths, was arranged primarily by columns, each data column containing detailed data relating to a position of the articulated truck, Table 3. For each truck position there were just three input cells:

Distance travelled from previous position
Steering lock increment to current point
Steering lock increment at this point

The user could enter values into these cells. The first of the three determined the distances between calculated (and plotted) positions, while the second and third described how the truck was steered. The data in the row labelled 'Steering lock leaving this point' were automatically calculated as the sum of 'Steering lock increment to current point' and 'Steering lock increment at this point'.

For each position the 'Inverse initial centre-line turning radius' and the 'Inverse final centre-line turning radius' were calculated by multiplying the 'Initial steering lock to current point' and the 'Final steering lock to current point' respectively by the 'Maximum inverse centre-line turning radius'.

SteerX and SteerY were used to calculate the X- and Y-ordinates of the centre point of the tractor rear axle group. RotatedX and RotatedY were used to calculate the new towing king pin coordinates from the previous king pin coordinates. TrailingX and TrailingY were used to calculate the new trailer rear axle group centre point coordinates. TrailingDirDeg was used to calculate the new trailer direction. At this stage the reference points that defined the position of the tractor and trailer had been calculated, and all other reference point coordinates could be calculated by means of RotatedX and RotatedY.

Truck- path plan view

The Paths sheet also contained a plan view of the loci of principal points on the tractor and trailer, Figure 4. The coordinates come from the main data rows of the sheet. A single cell on the sheet was designated to hold the current path position (within the range 1 to 250). A value could be manually entered into this cell. Cell blocks were also designated to hold the coordinates of the principal points on the tractor and trailer for the current path position. By means of the built-in spreadsheet function HLOOKUP (for looking up values in a horizontal cell range corresponding to a specified value within another horizontal cell range) these cell blocks were automatically populated with the data corresponding to the value in the position cell. In this way the truck outline was plotted automatically in any position when the corresponding position number was entered in the position box.

Little further work was required to automate the process of incrementing the value of the position variable when a button on the spreadsheet was clicked. Two procedures for the button's mouse-down and mouse-up events were used in conjunction with a simple time delay procedure called TimeTick (the code for these procedures is included in Appendix A). The resulting functionality was that the movement of the truck was animated: the truck advanced along its path when the Advance button was clicked down and continued to advance until the Advance button was released (a mouse-up event).

Using the truck-steering spreadsheet

In the example described here, the spreadsheet was used to investigate the clearance to the front and to the right that would be required for an articulated truck to exit a roadworthiness-testing lane and do a U-turn. It was straightforward to plot the exit opening. Using the built-in function MAX the north-most and east-most positions along the path line of the appropriate principal point on the articulated truck were found as the maximum Y- and X-ordinates of the path respectively. A cell was designated to hold the desired minimum approach clearance between the truck and the walls. Hence the positions of the front- and sidewalls were calculated, and the walls were plotted automatically on the plan-view diagram on the Paths sheet.

Many other situations could be examined using the approach that has been described here, for example, how various articulated trucks might negotiate a particular roundabout, turn onto a side road or steer around a series of obstacles.

Conclusion

This paper demonstrates the use of a spreadsheet application to solve engineering problems relating to the turning or steering of articulated trucks. The same techniques, or some of them, could be used in a wide variety of engineering situations. The approach and methodology could also be adapted readily to form the basis of a software application for analysing truck-steering problems. A search of the World Wide Web indicted that programs for this purpose do exist. However, this paper has shown that it is possible to turn an articulated truck on a spreadsheet.

Table 6 Part of the default truck position table on the Dimensions sheet.

Default Truck Position (Tractor Heading Straight in Y-direction with Rear Axle Group Centre Point at Origin)												
Trailer Angle	E		A		B		BL		BR		C	
Degrees	X	Y	X	Y	X	Y	X	Y	X	Y	X	Y
0	0.000	0.000	0.000	0.710	0.000	5.210	-1.245	5.210	1.245	5.210	0.000	-11.290
-35	0.000	0.000	0.000	0.710	0.000	5.210	-1.245	5.210	1.245	5.210	-6.883	-9.120

Figure 2 The trailer angle is 15°.

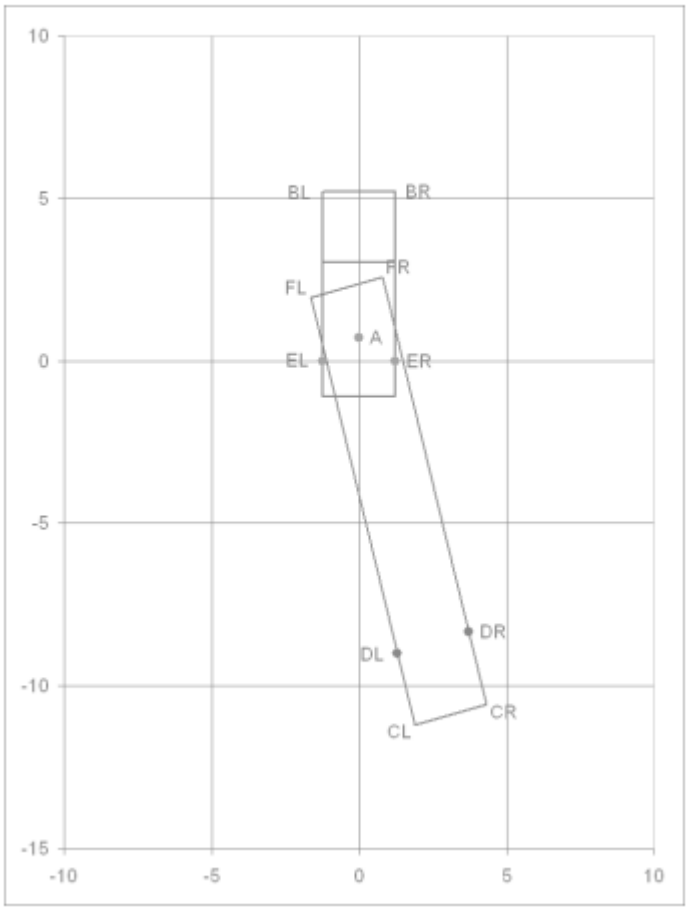


Figure 3 The trailer angle is 35°

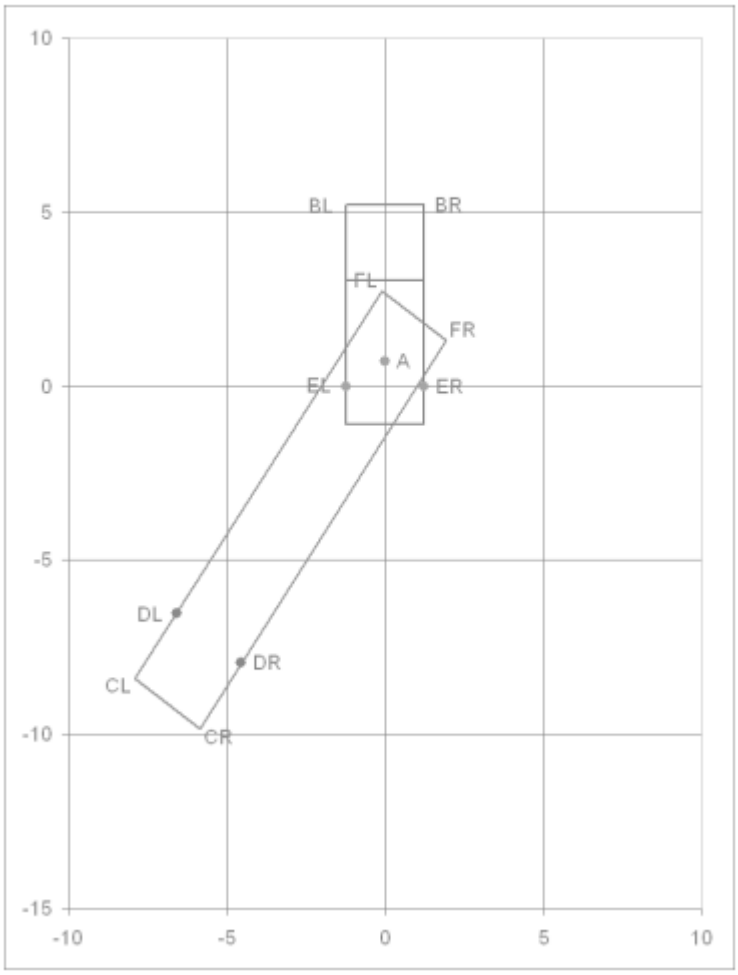


Figure 4 Diagram from the Paths sheet showing the path of an articulated truck to scale. The truck left a testing lane that had a specified width and did a U-turn. The closest front- and sidewall positions for a minimum approach clearance of 0.2 metres were also plotted. Pressing and holding a button on the sheet caused the truck to move on this diagram.

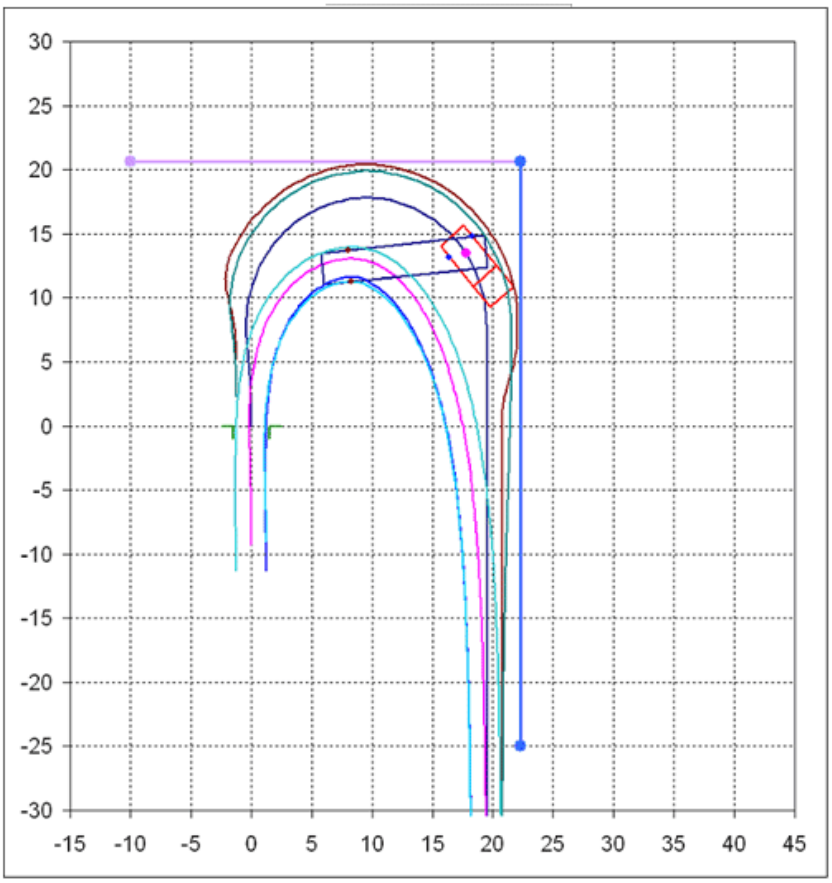


Figure 5 Illustration of a small movement of a trailing link.

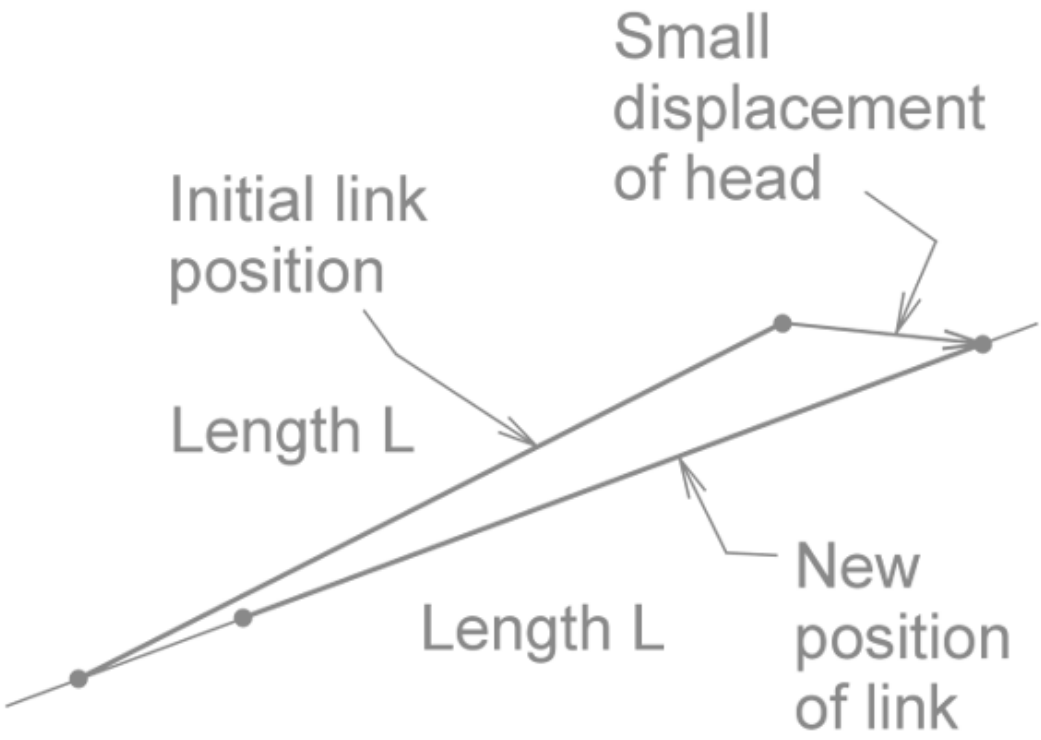


Table 1

Truck-defining dimensions (from the Dimensions sheet)	
Tractor width	2.490m
Steering axle width between contact centres	2.000m
Tractor cab length	2.170m
Tractor front to tractor front axle	1.410m
Tractor front axle to tractor rear axle group centre point	3.800m
Tractor overall length	6.295m
King pin to tractor front (length AB)	4.500m
Maximum steering angle	23.0°
Trailer width	2.600m
King pin to front of semi-trailer (length AF)	1.600m
King pin to rear of semi-trailer (length AC)	12.000m
Kin pin to trailer rear axle group centre point (length AD)	9.710m

Table 2

Dependent truck dimensions (from the Dimensions sheet)	
Overall length (tractor + trailer)	16.500m
Trailer swing radius	2.062m
Tractor rear axle group centre to trailer rear axle group centre (length ED)	9.000m
Tractor front to tractor rear axle group centre point (length BE)	5.210m
Tractor rear axle group centre point to tractor rear (length EJ)	1.085m
Tractor cab rear to tractor rear axle group centre point (length HE)	3.040m
King pin to tractor rear axle group centre point (length AE)	0.710m
King pin to cab rear (length AH)	2.330m
Tractor half width	1.245m
Trailer half width	1.300m
Steering axle half width	1.000m
Minimum clearance behind cab	0.730m
Minimum centre-line turning radius (at 100% steering lock)	9.952m
Minimum cab front corner turning radius	12.350m
Maximum inverse centre-line turning radius	0.10048 1/m

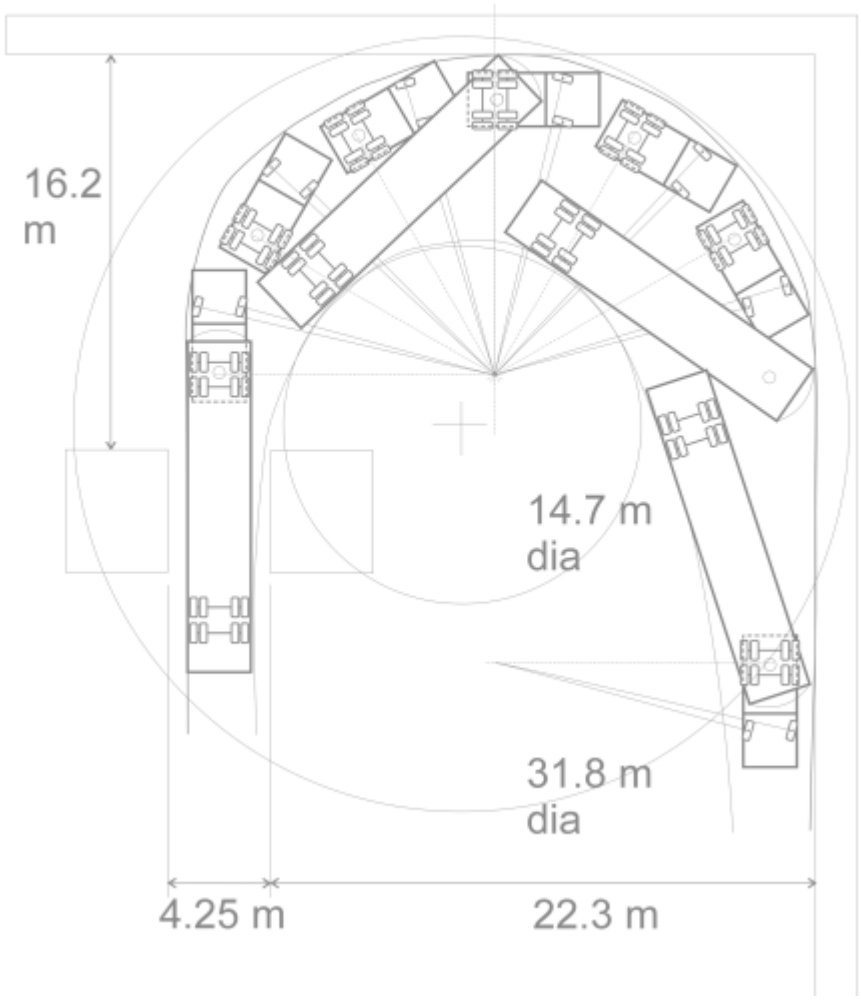
Table 3 Some of the data for the plotted path positions from the Paths sheet.

Position	1	2	3	4
Distance travelled from previous position (metre)	0.000	0.500	0.500	0.500
Initial steering lock to current point (%)	0.00	0.00	5.00	10.00
Steering lock increment to current point (%)	0.00	5.00	5.00	5.00
Final steering lock to current point (%)	0.00	5.00	10.00	15.00
Steering lock increment at this point (%)	0.00	0.00	0.00	0.00
Steering lock leaving this point (%)	0.00	5.00	10.00	15.00
Cumulative distance travelled (metre)	0.000	0.500	1.000	1.500
Current path input direction (degree)	90.00	90.00	90.07	90.28
Current path output direction (degree)	90.00	90.07	90.28	90.64
Inverse initial centre-line turning radius (1/metre)	0.00000	0.00000	0.00502	0.01005
Inverse final centre-line turning radius (1/metre)	0.00000	0.00502	0.01005	0.01507
Current centre-line turning radius (metre)	infinite	infinite	1.99E+02	9.95E+01
Path type	straight	curved	curved	curved
X-ordinate (centre point of tractor rear axle group) (metre)	0.000	0.000	–0.002	–0.006
Y-ordinate (centre point of tractor rear axle group) (metre)	0.000	0.500	1.000	1.500
King pin x-ordinate (metre)	0.000	0.000	–0.003	–0.009
King pin y-ordinate (metre)	0.710	1.210	1.710	2.210
Current cab front corner turning radius (metre)	infinite	infinite	2.00E+02	1.01E+02
Current centre X-ordinate (metre)		–203.107	–100.528	–66.795
Current centre Y-ordinate (metre)		0.250	0.500	0.750
Trailer rear axle group centre point X-ordinate (metre)	0.000	0.000	0.000	0.000
Trailer rear axle group pivot point Y-ordinate (metre)	–9.000	–9.210	–8.710	–8.210
Current trailer direction (degree)	90.00	90.00	90.01	90.02
Current trailer angle (degree)	0.00	0.00	–0.07	–0.26

Table 4 Final steering adjustments (from the Paths sheet)

Position	77	78	79	80	81	82	83
Distance travelled from previous position (metre)	0.500	0.500	0.500	0.500	0.500	0.500	0.500
Initial steering lock to current point (%)	–100.00	–95.50	–75.50	–55.50	–35.50	–15.50	0.00
Steering lock increment to current point (%)	4.50	20.00	20.00	20.00	20.00	15.50	0.00
Final steering lock to current point (%)	–95.50	–75.50	–55.50	–35.50	–15.50	0.00	0.00
Steering lock increment at this point (%)	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Steering lock leaving this point (%)	–95.50	–75.50	–55.50	–35.50	–15.50	0.00	0.00
Cumulative distance travelled (metres)	38.000	38.500	39.000	39.500	40.000	40.500	41.000
Current path input direction (degree)	279.48	276.66	274.19	272.30	270.99	270.25	270.02
Current path output direction (degree)	276.66	274.19	272.30	270.99	270.25	270.02	270.02
Inverse initial centre-line turning radius (1/metre)	–0.10048	–0.09596	–0.07586	–0.05577	–0.03567	–0.01557	0.00000
Inverse final centre-line turning radius (1/metre)	–0.09596	–0.07586	–0.05577	-0.03567	-0.01557	0.00000	0.00000
Current centre-line turning radius (metres)	–9.95E+00	–1.04E+01	–1.32E+01	–1.79E+01	–2.80E+01	–6.42E+01	infinite
Path type	curved	curved	curved	curved	curved	curved	straight

Figure 1 The steering axle wheel on the inside of a turn always makes a greater angle with the centre line of the tractor than the steering axle wheel on the outside of the turn



Appendix A

Full Listing of VBA Code

VBA CODE

Within the workbook, the custom functions and procedures were grouped within a module: these would have been accessible from any sheet within the workbook. There was also a small amount of code that was within the Paths sheet. This was associated with pressing and releasing the ‘Advance’ button on that sheet.

VBA Code Module

```
Public Sub Rotate(XInput, YInput, XCentre, YCentre, RotAngle, XNew,
YNew)

Pi = 4 * Atn(1)

X = XInput - XCentre
Y = YInput - YCentre

Radius = Sqr(X ^ 2 + Y ^ 2)

Select Case X

Case 0
   If Y < 0 Then
      Angle = 3 * Pi / 2
   Else
      Angle = Pi / 2
   End If

Case Else
   If Y < 0 Then
      If X < 0 Then
         Angle = Pi + Atn(Y / X)
      Else
         Angle = -Atn(-Y / X) + 2 * Pi
      End If
   Else
      If X < 0 Then
         Angle = -Atn(Y / -X) + Pi
      Else
         Angle = Atn(Y / X)
      End If
   End If
End Select

NewAngle = RotAngle + Angle

XNew = Radius * Cos(NewAngle) + XCentre
YNew = Radius * Sin(NewAngle) + YCentre

End Sub

Public Function RotatedX(XOrd, YOrd, XCentre, YCentre, AngleDeg)

Pi = 4 * Atn(1)
Call Rotate(XOrd, YOrd, XCentre, YCentre, AngleDeg * Pi / 180, XNew,
YNew)
RotatedX = XNew
```

```
End Function

Public Function RotatedY(XOrd, YOrd, XCentre, YCentre, AngleDeg)

Pi = 4 * Atn(1)

Call Rotate(XOrd, YOrd, XCentre, YCentre, AngleDeg * Pi / 180, XNew,
YNew)
RotatedY = YNew

End Function

Public Sub Trailing(LinkLen, HeadX, HeadY, HeadXPrev, HeadYPrev, _
TailXPrev, TailYPrev, TailX, TailY, DirDeg)

Dim Dist, Angle, NSteps, XStep, YStep, HeadXCurr, HeadYCurr, TailXCurr, _
TailYCurr, TailXCurrPrev, TailYCurrPrev, X, Y
Dist = Sqr((HeadY - HeadYPrev) ^ 2 + (HeadX - HeadXPrev) ^ 2)

Pi = 4 * Atn(1)

If Dist > 0.01 Then
   NSteps = Int(Dist / 0.01 + 0.5)
   XStep = (HeadX - HeadXPrev) / NSteps
   YStep = (HeadY - HeadYPrev) / NSteps
Else
   NSteps = 1
   XStep = (HeadX - HeadXPrev)
   YStep = (HeadY - HeadYPrev)
End If

HeadXCurr = HeadXPrev
HeadYCurr = HeadYPrev
TailXCurrPrev = TailXPrev
TailYCurrPrev = TailYPrev

For N = 1 To NSteps

HeadXCurr = HeadXCurr + XStep
HeadYCurr = HeadYCurr + YStep
TailXCurr = HeadXCurr - (HeadXCurr - TailXCurrPrev) * LinkLen / _
Sqr(((HeadYCurr - TailYCurrPrev) ^ 2 + (HeadXCurr - TailXCurrPrev) ^ 2))
TailYCurr = HeadYCurr - (HeadYCurr - TailYCurrPrev) * LinkLen / _
Sqr(((HeadYCurr - TailYCurrPrev) ^ 2 + (HeadXCurr - TailXCurrPrev) ^ 2))

TailXCurrPrev = TailXCurr
TailYCurrPrev = TailYCurr

Next
```

```
TailX = TailXCurr
TailY = TailYCurr

X = HeadX - TailX
Y = HeadY - TailY

Select Case X

Case 0
   If Y < 0 Then
      Angle = 3 * Pi / 2
   Else
      Angle = Pi / 2
   End If

Case Else
   If Y < 0 Then
      If X < 0 Then
         Angle = Pi + Atn(Y / X)
      Else
         Angle = -Atn(-Y / X) + 2 * Pi
      End If
   Else
      If X < 0 Then
         Angle = -Atn(Y / -X) + Pi
      Else
         Angle = Atn(Y / X)
      End If
   End If

End Select

DirDeg = 180 * Angle / Pi

End Sub

Public Sub RotationByRadius(Dist, InvRadius, XInput, YInput,
InputDirectionDeg, _
XCentre, YCentre, XNew, YNew, OutputDirectionDeg)

'RotAngle: angle of rotation in radians (used internally only)

Dim Pi, RotAngleDeg, frac

Pi = 4 * Atn(1)
InvRad = Abs(InvRadius)

If InvRad < 0.00000001 Then
   OutputDirectionDeg = InputDirectionDeg
```

```
    XNew = XInput + Dist * Cos(InputDirectionDeg * Pi / 180)
    YNew = YInput + Dist * Sin(InputDirectionDeg * Pi / 180)

    XCentre = ""
    YCentre = ""
Else
    If InvRadius < 0 Then
        RotAngleDeg = -(180 / Pi) * Dist * InvRad
    Else
        RotAngleDeg = (180 / Pi) * Dist * InvRad
    End If

    OutputDirectionDeg = (InputDirectionDeg + RotAngleDeg)

    Do While OutputDirectionDeg < 0
        OutputDirectionDeg = OutputDirectionDeg + 360
    Loop

    Do While OutputDirectionDeg >= 360
        OutputDirectionDeg = OutputDirectionDeg - 360
    Loop

    If InvRadius < 0 Then
        RadiusAngle = (InputDirectionDeg - 90) * Pi / 180
    Else
        RadiusAngle = (InputDirectionDeg + 90) * Pi / 180
    End If

    XCentre = XInput + Cos(RadiusAngle) / InvRad
    YCentre = YInput + Sin(RadiusAngle) / InvRad

    XNew = RotatedX(XInput, YInput, XCentre, YCentre, RotAngleDeg)
    YNew = RotatedY(XInput, YInput, XCentre, YCentre, RotAngleDeg)
End If

End Sub

Public Function RotationByRadiusX(Dist, InvRadius, XInput, YInput, _
InputDirectionDeg)

Call RotationByRadius(Dist, InvRadius, XInput, YInput, InputDirectionDeg, _
XCentre, YCentre, XNew, YNew, OutputDirectionDeg)
RotationByRadiusX = XNew

End Function

Public Function RotationByRadiusY(Dist, InvRadius, XInput, YInput, _
InputDirectionDeg)
```

```
Call RotationByRadius(Dist, InvRadius, XInput, YInput, InputDirectionDeg, _
XCentre, YCentre, XNew, YNew, OutputDirectionDeg)
RotationByRadiusY = YNew

End Function

Public Function RotationByRadiusXCentre(Dist, InvRadius, XInput, YInput,
_
InputDirectionDeg)

Call RotationByRadius(Dist, InvRadius, XInput, YInput, InputDirectionDeg, _
XCentre, YCentre, XNew, YNew, OutputDirectionDeg)
RotationByRadiusXCentre = XCentre

End Function

Public Function RotationByRadiusYCentre(Dist, InvRadius, XInput, YInput,
_
InputDirectionDeg)

Call RotationByRadius(Dist, InvRadius, XInput, YInput, InputDirectionDeg, _
XCentre, YCentre, XNew, YNew, OutputDirectionDeg)
RotationByRadiusYCentre = YCentre

End Function

Public Function RotationByRadiusOutDir(Dist, InvRadius, XInput, YInput, _
InputDirectionDeg)

Call RotationByRadius(Dist, InvRadius, XInput, YInput, InputDirectionDeg, _
XCentre, YCentre, XNew, YNew, OutputDirectionDeg)
RotationByRadiusOutDir = OutputDirectionDeg

End Function

Public Sub Steer(Dist, InvRadius1, InvRadius2, XInput, YInput,
InputDirectionDeg, _
XNew, YNew, XCentre, YCentre, OutputDirectionDeg)

'InvRadius: the inverse of the radius of rotation
'RotAngle: angle of rotation in radians

Dim DistInc, InvRadius, XIn, YIn, InDirDeg, XOut, YOut, XCen, YCen, _
OutDirDeg, XStep, YStep

If Dist > 0.01 Then
   NSteps = Int(Dist / 0.01 + 0.5)
Else
   NSteps = 1
End If
```

```
DistInc = Dist / NSteps
InvRad = InvRadius1
InvRadInc = (InvRadius2 - InvRadius1) / NSteps

XIn = XInput
YIn = YInput
InDirDeg = InputDirectionDeg

For N = 1 To NSteps
   Call RotationByRadius(DistInc, InvRad, XIn, YIn, InDirDeg, _
   XCentre, YCentre, XNew, YNew, OutputDirectionDeg)

   InvRad = InvRad + InvRadInc
   XIn = XNew
   YIn = YNew
   InDirDeg = OutputDirectionDeg
Next

End Sub

Public Function SteerX(Dist, InvRadius1, InvRadius2, XInput, YInput, _
InputDirectionDeg)

Call Steer(Dist, InvRadius1, InvRadius2, XInput, YInput, _
InputDirectionDeg, XNew, YNew, XCentre, YCentre, OutputDirectionDeg)
SteerX = XNew

End Function

Public Function SteerY(Dist, InvRadius1, InvRadius2, XInput, _
YInput, InputDirectionDeg)

Call Steer(Dist, InvRadius1, InvRadius2, XInput, YInput, _
InputDirectionDeg, XNew, YNew, XCentre, YCentre, OutputDirectionDeg)
SteerY = YNew

End Function

Public Function SteerXCentre(Dist, InvRadius1, InvRadius2, XInput, YInput,

_
InputDirectionDeg)

Call Steer(Dist, InvRadius1, InvRadius2, XInput, YInput, _
InputDirectionDeg, XNew, YNew, XCentre, YCentre, OutputDirectionDeg)
SteerXCentre = XCentre

End Function

Public Function SteerYCentre(Dist, InvRadius1, InvRadius2, XInput, YInput,
```

```
_
InputDirectionDeg)

Call Steer(Dist, InvRadius1, InvRadius2, XInput, YInput, _
InputDirectionDeg, XNew, YNew, XCentre, YCentre, OutputDirectionDeg)
SteerYCentre = YCentre

End Function

Public Function SteerOutDir(Dist, InvRadius1, InvRadius2, XInput, YInput, _
InputDirectionDeg)

Call Steer(Dist, InvRadius1, InvRadius2, XInput, YInput, _
InputDirectionDeg, XNew, YNew, XCentre, YCentre, OutputDirectionDeg)
SteerOutDir = OutputDirectionDeg

End Function

Public Sub SteerAndTrail(Dist, InvRadius1, InvRadius2, HeadXPrev,
HeadYPrev, _
TailXPrev, TailYPrev, InputDirectionDeg, LinkLen, HeadX, HeadY, TailX,
TailY, _
XCentre, YCentre, OutputDirectionDeg, TrailerDirectionDeg)

'InvRadius: the inverse of the radius of rotation

Dim Pi, DistStep, InpDirDeg, HeadXCurrPrev, HeadYCurrPrev, InvRInc

If Dist > 0.01 Then
   NSteps = Int(Dist / 0.05 + 0.5)
   DistStep = Dist / NSteps
Else
   NSteps = 1
   DistStep = Dist
End If

HeadXCurrPrev = HeadXPrev
HeadYCurrPrev = HeadYPrev
TailXCurrPrev = TailXPrev
TailYCurrPrev = TailYPrev
InpDirCurrDeg = InputDirectionDeg

InvRInc = (InvRadius2 - InvRadius1) / NSteps
InvRadius = InvRadius1

For N = 1 To NSteps

   Call RotationByRadius(DistStep, InvRadius, HeadXCurrPrev,
HeadYCurrPrev, _
   InpDirCurrDeg, XCentre, YCentre, HeadXCurr, HeadYCurr, _
```

```
    OutputDirectionDeg)
    Call Trailing(LinkLen, HeadXCurr, HeadYCurr, HeadXCurrPrev,
HeadYCurrPrev, _
    TailXCurrPrev, TailYCurrPrev, TailX, TailY, DirDeg)
    HeadXCurrPrev = HeadXCurr
    HeadYCurrPrev = HeadYCurr
    HeadXCurr = HeadXCurrPrev
    HeadYCurr = HeadYCurrPrev
    TailXPrev = TailX
    TailYPrev = TailY
    InpDirDeg = OutputDirectionDeg

    InvRadius = InvRadius + InvRInc
Next
HeadX = HeadXCurr
HeadY = HeadYCurr
TrailerDirectionDeg = DirDeg

End Sub

Public Function TrailingX(Dist, InvRadius1, InvRadius2, HeadXPrev,
HeadYPrev, _
TailXPrev, TailYPrev, InputDirectionDeg, LinkLen)

Call SteerAndTrail(Dist, InvRadius1, InvRadius2, HeadXPrev, HeadYPrev, _
TailXPrev, TailYPrev, InputDirectionDeg, LinkLen, HeadX, HeadY, TailX,
TailY, _
XCentre, YCentre, OutputDirectionDeg, TrailerDirectionDeg)
TrailingX = TailX

End Function

Public Function TrailingY(Dist, InvRadius1, InvRadius2, HeadXPrev,
HeadYPrev, _
TailXPrev, TailYPrev, InputDirectionDeg, LinkLen)

Call SteerAndTrail(Dist, InvRadius1, InvRadius2, HeadXPrev, HeadYPrev, _
TailXPrev, TailYPrev, InputDirectionDeg, LinkLen, HeadX, HeadY, TailX,
TailY, _
XCentre, YCentre, OutputDirectionDeg, TrailerDirectionDeg)
TrailingY = TailY

End Function

Public Function TrailingDirDeg(Dist, InvRadius1, InvRadius2, HeadXPrev, _
HeadYPrev, TailXPrev, TailYPrev, InputDirectionDeg, LinkLen)

Dim TrailDirDeg
Call SteerAndTrail(Dist, InvRadius1, InvRadius2, HeadXPrev, HeadYPrev, _
TailXPrev, TailYPrev, InputDirectionDeg, LinkLen, HeadX, HeadY, TailX,
```

```
TailY, _
XCentre, YCentre, OutputDirectionDeg, TrailDirDeg)

TrailingDirDeg = TrailDirDeg

End Function

Public Sub TimeTick()

StepTime = 0.1   ' Set duration.
Start = Timer    ' Set start time.
Do While Timer < Start + StepTime
   DoEvents     ' Yield to other processes.
Loop

End Sub
```

VBA Code on the Paths Sheet

```
Dim TimerOn As Boolean

Private Sub CommandButton1_MouseDown(ByVal Button As Integer, _
ByVal Shift As Integer, ByVal X As Single, ByVal Y As Single)
TimerOn = True
Do While TimerOn
   Range("Path_Position").Select
   Call TimeTick
   ActiveCell.Value = ((ActiveCell.Value) Mod 250) + 1
Loop
End Sub

Private Sub CommandButton1_MouseUp(ByVal Button As Integer, ByVal _
Shift As Integer, ByVal X As Single, ByVal Y As Single)
TimerOn = False
End Sub
```