



SPECTRA User Manual

Algorithms

Reference: SPCTRA_2.5/USER/ALGO/2.5
Issue Date: 13th February 1998
Author: Dave Skett

Approved by:

Project Manager:

Distribution:

Internal
Customers
Prospects

This document is the property of **Concept Systems Limited**. It must not be copied, in whole or in part, or otherwise disclosed, without prior written consent. Any copies of this document, or part thereof, must also include a copy of this legend. This document is supplied without liability for errors or omissions.

© **Copyright Concept Systems Limited 1998**

1 Logie Mill, Beaverbank Business Park, Logie Green Road, Edinburgh EH7 4HG, Scotland
Tel: (+44)131 557 5595 Fax: (+44)131 557 2367 Telex: 727673 CONCEP G

Table of contents

Chapter 0 - About this document.....	0-1
Revision history	0-1
Document cross-reference	0-1
Conventions used in this document	0-2
Chapter 1 - Source positioning.....	1-1
Choosing the model	1-1
The fixed model	1-1
The positioned model	1-1
Chapter 2 - NCN data processing	2-1
Overview	2-1
Preprocessing.....	2-1
Prediction	2-2
Adjustment	2-2
Chapter 3 - Streamer modelling.....	3-1
The default model.....	3-1
Modelling stretch.....	3-2
Chapter 4 - Streamer separations	4-1
Chapter 5 - Quality figures.....	5-1
Overall measures.....	5-1
Per-observation measures	5-1
HMP measures	5-1
Precision.....	5-1
Reliability.....	5-2
Chapter 6 - Observation SDs.....	6-1
Generalities	6-1
SD estimation.....	6-2
Chapter 7 - Shot prediction.....	7-1
The shot predictor	7-1
Lines and line-relative coordinates.....	7-1
Shooting modes.....	7-2

Conventions used in this document

The conventions used in this, and all other SPECTRA user manuals, are those defined in the *SPECTRA User Manual - Beginner's Guide*.

Chapter 1 - Source positioning

This chapter explains the gun models used for source positioning in SPECTRA.

Choosing the model

There are two sorts of model that can be applied to the guns, the *fixed* and the *positioned*. Which of these is applied is under control of the *General Survey Parameters* tab in the SCN Survey Parameters module (reached from the SCN main window by clicking the *Parameters* button). The setting is by default *Positioned*, and its state is saved in the SCN file.

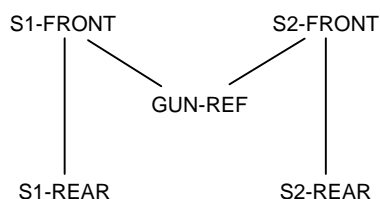
The fixed model

In the fixed model, the gun reference and all the nodes that refer to it are taken to make up a single rigid body. This rigid body, just like tailbuoys and vessels, contributes four quantities to the state vector: two position ordinates (of the gun reference) and two velocity ordinates. The attitude of the body is the attitude of the towing vessel, i.e. what is given by that vessel's gyro. In order to compute O-Cs for the observations (ranges, bearings, etc) of the various nodes on the body, we need to be able to compute the positions of the various nodes. The position of the gun reference is taken directly from the state vector; the position of any other node is computed from that by using the nominal offsets, rotated by attitude, as a layback from the gun reference position.

The problem with this model is, of course, that the gun strings are by no means rigidly attached to each other. The advantage of it is that it needs few states, and thus allows some redundancy. This model can work better in some 2D jobs where, for example, there are only laser range/bearing pairs to each string (i.e. unlike 3D jobs there are no acoustics to the guns).

The positioned model

The positioned model is best illustrated by referring to this example:



In this picture GUN-REF is a gun reference node, and the other nodes are all non-reference nodes that refer to GUN-REF.

This model is somewhat more complex than the fixed one. The nodes that reference the gun reference are divided into groups of (roughly) equal nominal x offset – in the example into the groups {S1-FRONT, S1-REAR} and {S2-FRONT, S2-REAR}. Each of these groups, or “strings”, is treated as a rigid body, and again the attitude is that of the towing vessel. Each string contributes two states to the state vector: the position of the forward-most node on the string, i.e. the string with smallest (in absolute value) y offset – S1-FRONT and S2-FRONT in the example. The strings are taken to have a common velocity, i.e. the entire assembly contributes two velocity states to the state vector. Thus, in the example there is a total of 6 states for the new model: two position states for S1-FRONT, two position states for S2-FRONT and two velocity states for the whole assembly. In the fixed model there would have been four states in total.

For output purposes only there is another aspect to the positioned model. This is that the position that is output for the gun reference is, in fact, a weighted average.

The SCN allows a weight to be defined for gun reference nodes and nodes that refer to them. Being weights these must be non-negative; internally the NCN sums up the weights and divides each by the total to get the weighting factor that is applied, so that giving each node a weight of 1 is the same as giving each a weight of 10. As a special case, if all of the weights are zero the NCN sets them all equal to 1 prior to the normalisation step above.

Since the gun string positions are in the state, the NCN computes their positions directly from the observations. During the output stage the position of the gun array reference is computed. For each node on each string, an array reference position is computed by using the node position and the negative of its offset. The array reference position that is output is the weighted average of all these array reference positions, the weights being those entered in the SCN. The error ellipse for the output array reference positions is the weighted average.

Note that it is the average of the various strings’ ideas of where the reference node is that are averaged, not the string positions themselves.

Suppose in the example we had these offsets:

	X	Y
S1-FRONT	-10	5
S1-REAR	-10	-15
S2-FRONT	15	5
S2-REAR	15	-15

And that at some point the positions for the strings were estimated as:

	E	N
S1-FRONT	100	30
S1-REAR	100	10
S2-FRONT	130	31
S2-REAR	130	11

If all the weights were equal, the computed reference position would be:

$$\text{East: } (100--10)/4 + (100--10)/4 + (130-15)/4 + (130-15)/4 = 112.5$$

$$\text{North: } (30-5)/4 + (10--15)/4 + (31-5)/4 + (11--15)/4 = 25.5$$

If the weights on the S1-FRONT and S1-REAR were both 1.0 and those on S2-FRONT and S2-REAR were 0.0, the computed reference would be:

$$\text{East: } (100--10)/2 + (100--10)/2 = 110.0$$

$$\text{North: } (30-5)/2 + (10--15)/2 = 25.0$$

Chapter 2 - NCN data processing

This chapter describes the data processing algorithms employed by the Network Calculation Node.

Overview

The SPECTRA Network Calculation Node is responsible for performing network adjustment. Each instance of the NCN runs a Kalman filter for a specific network. There are always at least two NCNs: one running the shot predictions and the other the “primary” network, i.e. the network which is logged in the P1 and P2 files.

Leaving aside such details as configuration and reconfiguration, the primary NCN operates as follows:

```
Loop:
  Check for shot.
  If found: predict state at time of shot, and output.
  Check for observation data.
  If found:
    Sort the observation data by time.
    For each time T:
      Preprocess the observations at T.
      Predict state to time T, and update filter state and time.
      Adjust state using observations.
  Sleep
  Goto Loop
```

Note that the state is **not** updated as part of the shot output. It is possible, but *not* problematical, that a shot at time T, say, is processed after observations that are at times later than T.

Other NCNs operate similarly except that they may output each time observations are processed, rather than just at shot.

All the matrix computations described below are to be taken as meaning that the whole matrix on the left hand side is updated. In particular a full state error covariance matrix is used. As a matter of implementation, we take advantage of sparsity – of the observation model matrix and the state transition matrix – to reduce computational burden; this is purely an optimisation: we get the same answers as if we had treated the matrix in question as being full (mostly of zeroes).

Preprocessing

In this phase fixed offsets and scales are applied to the observation data, and, if gating is enabled, then observations that fail the gate are so marked.

Prediction

In this phase the filter state (\underline{x}) and state error covariance matrix (P) are updated. Given the time interval between filter time and time to predict to, the NCN computes the state transition matrix (F) and driving noise covariance matrix (Q) and then computes the predicted state ($\hat{\underline{x}}$) and predicted state error covariance matrix (\hat{P}) via:

$$\hat{P} = F * P * F' + Q \quad (F' \text{ is the transpose of } F)$$

$$\hat{\underline{x}} = F * \underline{x}$$

Adjustment

In this phase we have a vector (\underline{o}) of observations and an observation noise covariance matrix (G), and use these to produce an updated state estimate and state error covariance estimate. By default the NCN processes the observation vector as a whole, but an option is available to make it treat the observations one at a time.

The observations are treated as being not only noisy, but fallible; that is observations may be rejected.

In the description below, the “observation sized” vectors and matrices (i.e. $\underline{o} - c$, A , G , D and R) are to be taken as containing only the observations that have not yet been rejected. Initially only those observations that were gated in preprocessing are marked as rejected (this could perhaps be none).

The computations are:

1. Compute $\underline{o} - c$, the “innovations” vector (i.e. the \underline{o} vector less the vector of computed observations, computed from the predicted state), and A , the observation model matrix (i.e. the matrix of derivatives of the calculated observations with respect to the state).
2. Compute D , the innovations covariance matrix, via $D = G + A * P * A'$, and its inverse R .
3. Compute the “V” statistic, $V = \underline{o} - c' * R * \underline{o} - c$. If this is less than its critical value, go to step 4, otherwise perform “data-snooping”:
 - for each observation i :
Compute $\underline{W}[i] = (R * \underline{o} - c)[i] \div \sqrt{R[i][i]}$
 - If the maximum $\underline{W}[i]$, $\underline{W}[i0]$ say, exceeds the critical value for the W test, mark observation $i0$ as rejected and go to step 1; otherwise go to step 4.
4. Update \underline{x} via
 $\underline{x} = \underline{x} + P * A' * R * \underline{o} - c$
5. Update P via
 $P = P - P * A' * R * A * P$

6. Compute the residual vector (i.e. the \underline{o} vector less the vector of computed observations, computed from the adjusted state). Note that since the adjusted state will differ from the predicted state, the residual vector will be different from the $\underline{o} - \underline{c}$ vector.

The implementation differs from the above description due to optimisations. These are purely optimisations; the algorithm is as described.

The critical values for the V test and for data-snooping are both under user control. Moreover, there is an option to unlink the V test and data-snooping, i.e. to always perform data-snooping, whether or not the V-test passed.

Chapter 3 - Streamer modelling

This chapter explains the streamer modelling used by SPECTRA.

The default model

The default streamer model is parameterised by:

- a position (of the streamer reference);
- a number (roughly, two less than the number of compasses) of streamer tangent azimuths, one of which is at the reference position. These are the streamer tangent nodes.

This model consists of supposing that the streamer shape is that of a number (1 less than the number of tangent azimuths) of circular arcs, with abutting arcs having the same tangent where they abut. It allows the computation of the position of any point down the streamer, as functions of the parameters, given the offset of the point in question from the streamer reference.

If one knows the position of one endpoint of an arc, the azimuth ($A1$) of the tangent at the known point, the azimuth ($A2$) of the tangent at the other endpoint, and the arc length (L), then the position of the other endpoint can be computed as the position a distance D and bearing B from the known position, where:

$$D = L * Sinc(0.5 * (A2 - A1))$$

(where $Sinc(x) = \sin(x) \div x$, for x measured in radians)

$$B = 0.5 * (A1 + A2)$$

The position of the streamer reference is in the state. The position of the tangent node immediately forward of the reference point can then be computed as above, because the tangent azimuths are in the state and the arc length is the absolute difference of the offsets. Given this new position, the position of the tangent node forward of it can then be computed by the same method. The tangent nodes aft of the reference node are computed in the same way, by stepping aft along the streamer

To find a position other than a tangent node position (for example, an acoustic node position), first of all the tangent node nearest to it is found and the arc length between the wanted position and the tangent node position is computed. The tangent azimuth of the arc at the wanted position is determined (by linearly interpolating between the bounding tangent node azimuths) and so, as above, the wanted position is computed.

Note:

For the purposes of observation equations and of computing error ellipses, “as functions of the parameters” means not only being able to compute the positions given the parameter values, but also being able to compute the derivatives of those positions with respect to the parameters.

Modelling stretch

The NCN command line parameter `-STCH` modifies this model by adding another model parameter to the state: the total stretch of the streamer.

The model of the streamer stretch is that the local stretch factor is a linear function of offset.

Local stretch factor: That function of unstretched offset such that the stretched arc length between two points is the integral of the function between the offsets of the points, with respect to offset.

This follows from the assumptions that the load on the streamer is a linear function of offset and that the streamer obeys Hooke's law (with a constant spring constant!).

If the ratio of loads at the tail and head of the streamer is known, then the two coefficients of the stretch factor function can be expressed as (linear) functions of the total stretch. At present the NCN makes the simplifying assumption that the load on the tail (i.e. due to the tailbuoy) is zero; this means that the coefficients of the stretch factor are independent of the loads. This assumption will shortly be removed; it was made not for reasons of algorithmic complexity within the NCN, but rather for interfacing reasons (i.e. identifying where to find the load data).

Given the stretch factor function, the modifications to the position (and derivative) computations amount to using the stretched lengths instead of difference of offsets, and to computing the derivatives of position with respect to overall stretch.

Explicitly, in the formula for D above, L (which is there the absolute difference of the offsets) is replaced by the stretched length – i.e. the integral of the local stretch factor between the two offsets, with respect to offset.

Note that if the offsets given in the NCN have already been adjusted for expected stretch, then the `-STCH` flag can still be applied: the computed stretch will then represent a correction to the expected stretch.

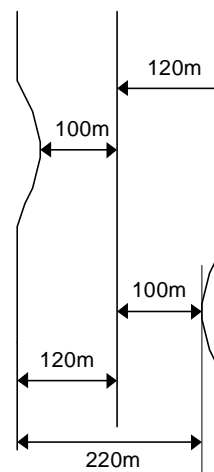
Chapter 4 - Streamer separations

This chapter describes the streamer separation calculations used by SPECTRA.

Streamer separations are computed between pairs of streamers. These separations are: the closest approach of the forward-most third of the streamers, of the middle thirds, of the aft-most thirds, and the overall closest approach. These figures are in metres, and, being distances, are never negative.

Note that this means that the various separations do **not** add; i.e. if the separation of streamer1 from streamer2 is 100m and of streamer2 from streamer3 is 100m it is not necessarily the case that the separation of streamer3 from streamer1 is 200m. The various minimum distances could well occur at different offsets.

For example:



In this case the separation of streamer 3 from streamer 1 will be 220m.

The algorithm uses the receiver group positions. These are computed (for purposes of P1 logging) from the streamer model.

Given these, the algorithm treats the streamer thirds separately, and each third as consisting of line segments between adjacent receiver group positions. For each pair of thirds (i.e. both forward-most thirds, both middle thirds and both aft-most thirds), it proceeds in two passes.

In the first pass, for each receiver group in the first streamer third, the closest point on the second streamer third (taken to be a sequence of line segments) is found (basically by searching), and the distance between the receiver group and the closest point is computed. The least such distance, over all receiver groups in the first streamer third, is recorded.

The second pass is the same except that the roles of the two streamers are exchanged. The lesser of the two recorded distances is then taken as the separation between those thirds.

The overall separation of the streamers is taken to be the least of the separations between the thirds.

The selection of which pairs of streamers to compute separations between is under user control, in the Display Node. Note that the above algorithm assumes that the cables are, more or less, alongside one another. In particular they will not give useful answers in the case that one streamer is aft of another!

It might be objected that we should be using arcs between adjacent receiver group positions rather than line segments. However, the differences are negligible. Consider, for example, a 4km streamer with 160 receiver groups that had a 90 (!) degree feather. The tangents of the cable at adjacent receiver group positions will then differ, on average, by 90/160 degrees, and the arc length will be 25m. The greatest distance from the arc to the chord is then:

$$d = (25 / A) * (1 - \cos(A / 2))$$

where A is $(90/160) * \pi/180$ radians, or roughly 0.01 radians, this gives 0.031m.

In comparison with the error ellipse size (i.e. in comparison with the uncertainty in the receiver group positions), this is negligible.

Chapter 5 - Quality figures

This chapter describes the measures of quality used by SPECTRA.

Overall measures

<i>Degrees of freedom</i>	Two different figures are given for degrees of freedom. One is the number of instances of observations processed since the last output. The other is the percentage of observations that have been processed.
<i>Quality</i>	Sum of the V statistics (see “ Adjustment ” on page 2-2) for all adjustments since the last output, divided by the degrees of freedom.

Per-observation measures

- The o_c and residual vectors.
- The vector of Ws (as in “[Adjustment](#)” on page 2-2), also known as the normalised residuals.
- The vector of marginally detectable errors.

Note that for any particular observable, e.g. a particular compass, it is the last processed instance of it that appears in the vectors above, and so the different components of the vectors may refer to different times. The vector of times is also output.

HMP measures

The HMPs computed are the midpoints between the source position(s) and a number (typically two less than the number of compasses) of roughly equally spaced points down each streamer. The selected points are, in fact, the streamer tangent node positions.

For each such HMP we compute the derivatives – with respect to the state vector – of its position coordinates, in a matrix H . H is a $2 \times S$ matrix, where S is the dimension of the state vector.

Precision

We compute the HMP covariance matrix C via $C = H * P * H'$ (where P is the state error covariance matrix).

From C we compute the HMP uncertainty figures: the semi-major axis (the largest eigenvalue of C) and the 2D rms (the square root of the trace of C).

The largest uncertainties over all HMPs (which may be for different HMPs) are output for display and logging.

Reliability

For each observation in the network we compute the shift due to a “that-observation’s-MDE” sized innovation, i.e. we compute the vector

$$\underline{dx} = P * \underline{A}' * R * MDE$$

where P is the state error covariance matrix, \underline{A} is the observation model matrix, and R is the inverse of the innovations covariance matrix ($G + \underline{A} * P * \underline{A}'$, where G is the observation noise covariance matrix). Since we deal here with one observation at a time R and G are scalars and \underline{A} is a vector.

From \underline{dx} we compute the shift \underline{dh} in the HMP position: $\underline{dh} = H * \underline{dx}$. From \underline{dh} we compute the radial shift (i.e. the length of \underline{dh}), the along-line shift (i.e. the absolute value of the component of \underline{dh} in the along-line direction – note that HMP measures are only computed when on-line), and the cross-line shift. The largest of each (over all observations and all HMPs) are output for display and logging.

Chapter 6 - Observation SDs

This chapter discusses observation noise standard deviations in SPECTRA.

Generalities

As with all least squares systems the NCN requires estimates of the amount of noise on each observation. The “amount” of noise is measured by the standard deviation of the noise.

An important point is that the SD is that of the noise, not of the signal – an observation may vary rapidly because that is the nature of what is being measured, and still have a small noise SD.

The observation SDs are used for two related, but distinct, purposes:

- To weight the observations, with the weight of an observation being one over the SD squared. This means that less reliance is placed upon noisier observations.
- To decide when an observation instance should be rejected, that is when an observation value represents a spike that should not be used in processing. A rough rule of thumb is that observation values more than 4 SDs away from their computed values are rejected, (although in fact the uncertainty in the computed value is also taken into account) so that the larger an observation SD is, the less often it will be rejected.

In pure least squares systems little changes if all of the observation SDs are scaled by the same factor – in particular the computed states and observations are unchanged. However, this is not the case for Kalman filters for they are tuned not only by observation SDs, but also by the driving noise “SDs”.

The accuracy of the SD figures being used in a filter can be estimated by looking at the quality figures. If all of the SDs used are perfectly accurate, and there are no modelling errors, then the Network Quality figure will have mean 1, and each observation normalised residual time series will have mean 0 and SD 1. Moreover, if a particular observation is too low (or too high) then the SD of its normalised residual time series will be greater than (or less than) 1.

However, in actual practice it is rare that there are no modelling errors. These can arise in many ways. For example, a node position might be mis-specified (e.g. its offset from another node given incorrectly), or a sensor mis-calibrated, or a scale factor (e.g. speed of sound in water) might be given incorrectly.

There are also more subtle, and far less easily corrected, modelling errors:

- It may be that observations that are treated as having independent noises on them are in fact somewhat correlated (e.g. rGPS range bearings)
- It may also be that, while the model assumes “white” noise, (i.e. that the noise on an observation is uncorrelated with the noise on the previous instance of that observation) in fact the noise is “coloured”. This happens when the observation has been filtered – compasses are an example.

Modelling errors lead to observation normalised-residual (and plain residual, and O-C) time series having non-zero mean, i.e. (some of) the observations appear, to the NCN, to be biased.

While it is true that in a network without modelling errors it is best to choose the most accurate SDs possible, in the presence of modelling errors the situation can be more delicate.

For example, suppose you have an acoustic range with a 0.5 metre bias, whose noise SD was 0.1 metres. If you were to use that 0.1 metre SD, then it is likely that many, perhaps all, instances of the observation would be rejected and you would gain little or no benefit from the range. On the other hand if you used, say, 0.25 metres for the SD, then you would gain benefit from this range.

The moral of this story is that you must take observation bias into account when setting SDs, and that you may get better filter performance by using artificially large SDs for biased observations, even though this will result in sub-optimal quality figures.

SD estimation

Starting with version 2.4, the NCN supports estimation of observation SDs and use of the estimated SDs. These options are independently controlled by command line options, i.e. you can have the NCN just estimate, or estimate and use the estimated values.

When the estimation of observation SDs is enabled, the NCN runs a separate SD estimation filter for each observation. These filters are fed only with raw data; they are not aware of calculated values. Only observation instances that were neither gated nor rejected are fed to these filters, so that their SD estimates are not polluted by spikes.

When the use of estimated SDs is enabled, the estimated SDs are used unless the estimated value is less than: either a floor value (20% of the nominal) – in which case the nominal value is used; or the (recent) bias on the residual time series for that observation – in which case the value used is the square root of the sum of the bias squared and the estimated SD squared. Note that if the floor values are set too high (i.e. are higher than the observed noise SDs) there will be little benefit in using estimated SDs, for they will all be stuck at their floor values.

It is hoped that the use of estimated SDs will lead to an increased diagnostic power of the quality figures.

If the system is without modelling errors then the quality figures will be driven to their ideal values. That is, when using estimated SDs, the fact that the Network Quality is not 1 (and correspondingly that not all of the normalised residual time series have SD of 1) will be due to biases preventing the SDs from being set low enough.

Chapter 7 - Shot prediction

This chapter discusses shot prediction in SPECTRA.

The shot predictor

The shot predictor is the part of the NCN that controls shot prediction. It is only active in the ncc NCN. It is fed with NETSOLUTIONs and produces estimates of the time when the next shot should be fired. In itself the shot predictor is wholly deterministic: it regards the positions and velocities in the NETSOLUTION as being accurate, and does no filtering of any kind. There are two kinds of output from the shot predictor: the SHOTPREDICT item, which specifies when the next shot should be fired; and the LINESTATUS item. The SHOTPREDICT item is read by the RTNU; the LINESTATUS item is read by SPECTRA Nodes that care about where you are on line (i.e. on approach, on line, run-out, off line).

Lines and line-relative coordinates

SPECTRA supports three sorts of line: great circle, grid, and grid true. The first are the paths of shortest distance between their endpoints, while the latter two are both straight lines in the given projection. It is worth remembering that in both cases the line azimuth will vary (slowly) along the line, and that, in SPECTRA, we always deal with azimuths relative to **true** north, never map north.

Line-relative coordinates are computed similarly in the two cases: given a point we determine that point on the line such that the vector to the given point from the point on the line is at right angles to the line at that point, i.e. we “drop a normal to the line”. The dc (“distance cross”) coordinate is the length of this vector, while the da (“distance along”) is the distance along the line from the start of the line to the foot of the normal. Note that on great circle lines and on grid true lines the distances in question are **true** distances, while on grid lines they are **map** distances, uncorrected for scale. Line relative velocities are computed by resolving the velocity vector along directions parallel to (for valong) and normal to (for vcross) the line, at the foot of the normal.

Preplot points are computed assuming a constant along difference between successive preplots. This means that: on great circle and grid true lines successive preplots are a constant **true** distance apart; on grid lines successive preplots are a constant **map** distance apart, and hence a slowly varying **true** distance apart.

Shooting modes

SPECTRA supports three shooting modes, *along track*, *time* and *cycle test*. Of these, along track is by far the most commonly used. Cycle test, as its name suggests, is used mostly for causing the system to cycle (i.e. produce shot predictions) for testing purposes.

Along track

In along track shooting, the shots are to occur when the dalong of the shooting point is the same as that of the preplot. The shooting point is a point a given distance along the line from the ncc network reference point (the distance being given by the "Shot Layback" field from the LMN). The shot predictor monitors the shooting point dalong and valong, computes from this when the shooting point's dalong will be that of the preplot and publishes a SHOTPREDICTION with the computed time. SHOTPREDICTIONS are published whenever the ncc NCN processes raw data, and (for the next shot) whenever a SHOT event occurs. Note that for this to be possible the valong of the shooting point must be positive; indeed when the valong is less than 0.1 Knots the shot predictor will not issue predictions, and the LINESTATUS will be set to off line. The same situation holds when the shooting point is too far (more that 50km) off line.

Cycle test

In cycle test mode, the shot predictor issues a stream of predictions whose predicted times are the given number of seconds apart, however far the shooting point is from the line and whatever its velocity is.

Time

Time shooting is a mixture of along track and cycle test. Along track shooting is used when in lead-in, and cycle test (i.e. timed) shooting is used in line and in run-out.