Australian Government
Department of Defence
Defence Science and Technology Organisation

Numerical Calculations for Passive Geolocation Scenarios

Don Koks

Electronic Warfare and Radar Division
Defence Science and Technology Organisation

DSTO–RR–0319

ABSTRACT

This report reviews work done in gaining some familiarity with methods of passive geolocation, and a search for rules of thumb that might tell us how to optimise the geolocation for a given scenario. We first cover the main approaches to collecting angle of arrival data and point out typical accuracies. Following this is an account of the mathematics used to analyse this data to produce an estimate of an emitter's location. We then give an overview of some of the literature, and finish by demonstrating a Matlab programme that runs several geolocation algorithms. Simple rules of thumb that specify how to fly a baseline and take data, so as to maximise the accuracies of the different techniques, do not appear to be readily derivable.

Published by

Defence Science and Technology Organisation
PO Box 1500
Edinburgh, SA 5111, Australia

Telephone: (08) 8259 5555
Facsimile: (08) 8259 6567

AR No. AR–013-779
January, 2007

Numerical Calculations for Passive Geolocation Scenarios

EXECUTIVE SUMMARY

This report has been written with two broad aims. The first is to present an overview of current geolocation techniques when bearings-only data is given. We give a short overview of geolocation literature, as well as covering the backgrounds to several techniques: the Cartesian Pseudo-Linear Estimator (which we extend to handle the case of a moving emitter), the Gauss–Newton method, Recursive Least Squares, and two types of Kalman Filter. These discussions are given with a view to being sufficient to enable the reader to write code that implements these routines, as well as understanding some of the relevant theory.

The second aim of this report is to analyse representative geolocation scenarios, in an effort to quantify how well each technique can be expected to perform. We do this by using a Monte Carlo approach of repeated runs using software written in Matlab specifically for this research. We discuss a search for rules of thumb that could direct how to optimise the geolocation geometry for a scenario involving one emitter and a constant velocity receiver. In this and other contexts, the Cramér–Rao bound is discussed in some detail. This bound gives the theoretical limit on the best accuracy that a geolocation approach can attain.

The results of analysing representative geolocation scenarios show that no straightforward rules of thumb appear to be derivable; it seems that nontrivial geolocation scenarios contain too many parameters to allow each of the techniques' geolocation accuracies to be readily derivable using a simple rule. The situation only gets worse for more complex scenarios involving multiple moving receivers and a single, perhaps moving, emitter. Multiple emitters are not handled by this report; their presence calls for more specialised techniques.

Author

Don Koks
Electronic Warfare and Radar Division

Don Koks completed a doctorate in mathematical physics at Adelaide University in 1996, with a dissertation describing the use of quantum statistical methods to analyse decoherence, entropy and thermal radiance in both the early universe and black hole theory. He holds a Bachelor of Science from the University of Auckland in pure and applied mathematics, and a Master of Science in physics from the same university with a thesis in applied accelerator physics (proton-induced X ray and γ ray emission for trace element analysis). He has worked on the accelerator mass spectrometry programme at the Australian National University in Canberra, as well as in commercial internet development. Currently he is a Research Scientist with the Maritime Systems group in the Electronic Warfare and Radar Division at DSTO, specialising in jamming, three-dimensional rotations, and geolocation.

Contents

Appendices

Glossary, Conventions, and Constants

CEKF Cartesian Extended Kalman Filter, a geolocation algorithm.

CEP Circular Error Probable. The radius of a circle drawn around the emitter that encompasses 50% of all estimates made of its position. Thus 50% of the time, the emitter will lie somewhere within a circle drawn around the estimate. In other words, we can be 50% confident that the emitter lies within the CEP distance of any particular estimate we make. Increasing this parameter to a higher certainty, say 95%, can increase the corresponding CEP figure greatly.

DF Direction Finding.

DSTO Defence Science Technology Organisation.

G–N Gauss–Newton geolocation algorithm.

MPEKF Modified Polar Extended Kalman Filter, a geolocation algorithm.

CPLE Cartesian Pseudo-Linear Estimator geolocation algorithm.

RLS Recursive Least Squares geolocation algorithm.

TDOA Time Difference Of Arrival (also known as DTOA, Differential Time Of Arrival).

1 Overview

Passive geolocation in a broad sense has two rather separate parts. The starting point for locating an emitter is the direction-finding process (DF), that is based upon hardware and signal reception. Analysis then follows on the observed data; and in a noisy world, this draws heavily on a statistical approach.

Direction-finding techniques make up a set of possibly complementary tools that are used to produce bearing data suitable for statistical analysis. A general description of these is presented in Section 2 of this report. Throughout the remaining sections we concentrate more on the analysis of the bearings that the DF system provides, taking the provision of (noisy) bearings as a given. Analysis of this bearing data itself is described in Section 3. There we discuss several different methods, all of which are aimed at producing an estimate of where the emitter actually is and how fast it might be moving.

Although geolocation has a long history, much good work in the field has been done only relatively recently, as computer power continues to increase. The various techniques that in the past were graded on their computational intensity are now all performed easily at high speed on a modern computer, and the question of which one is to be preferred is partly a question of how easy they each are to implement and understand. We discuss recent work in the various geolocation techniques in Section 4.

Section 5 deals with a very specific scenario where three fixed receivers geolocate one stationary emitter. We produce Cramér–Rao bounds, as well as comparing the results of different geolocation routines. Long-baseline TDOA is then discussed for this geometry, with results presented that give an idea of how the geolocation is affected by baseline length and timing accuracy.

Since one of the aims of the current work has been to develop a feel for the types of paths that must be flown to achieve a given geolocation accuracy, we need to be able to run the various bearing analysis routines on sample data to see what effects the various parameters have on the final estimate of the emitter's position and state of motion. To this end, in Appendix C we show the results of a Matlab programme that runs these routines as required. Section 6 analyses these results with a view to obtaining some rules of thumb for simple scenarios.

2 Direction-Finding Techniques

The art of direction finding itself goes back to Hertz and the beginnings of radio theory, when it was realised that the anisotropy in an antenna's reception pattern could be used to find an emitter's direction. This is still a common means of direction finding, but has long been supplemented by more sophisticated coherent techniques. Because of this, it is convenient to split DF apparatus into two classes: those that make use of the receiver's amplitude response, and those that use its phase response. The amplitude response type can be of small size and thus fit well into fighter aircraft; on the other hand, the accurate measurement of phase differences in the arrival of a wave front can require a larger receiver, and antenna array, than might be practical on these.

Even if we are able to locate an emitter to any desired accuracy, our measurements can still be limited by our knowledge of our own ship's position and heading. However, own-ship position and heading are determined very accurately using GPS/INS.

2.1 Amplitude Response

The first of the noncoherent DF techniques is the simplest and oldest: locating the direction of maximum signal strength while rotating the antenna. This has the simplifying advantage that the only detailed knowledge of the beam pattern needed is the beam width, where a typical resolution of this type of system is about 1/10 of that width. But this simplicity brings several possible problems:

- The receiver might have to be held wide open to maximise signal reception probability, which is a disadvantage in the presence of many signals. On the other hand, this sort of wide band DF is not always required, with narrow band searching instead being favoured. In some situations a wide band scan might still be done, the signal located and then further searching done over a much smaller bandwidth.
- Many pulses need to be intercepted—with a correspondingly large amount of data processed.
- Signals of varying strength can completely distort the measurement. This is the main problem and can lead to the method being effectively useless, unless a correction is applied. Normally this is overcome by employing an extra nonscanning nondirectional antenna that also processes the pulses, allowing us to normalise the data by comparing the ratios of the signal strengths seen by each antenna.

Alternatively, we can dispense with the rotation by using at least two stationary antennas with known, nonisotropic beam patterns: overlapping these patterns and comparing the signal strength from each then gives the source direction, even when the signal strength is changing. This method of amplitude comparison is used very commonly for direction finding. Typically there are four or more antennas, with the coarse tuning being done by simply measuring which has the strongest signal, followed by fine tuning by comparing responses as above. Small errors in power received can lead to a large DF error, so that typical bearing accuracy for four receivers is 10–15°. This can be increased to about 5° for six receivers, while 2° is attainable with eight receivers.

2.2 Time Difference of Arrival

If our platform is large enough, measuring the difference in arrival times of the signal at two or more widely spaced receivers will give good DF information. This technique is known as Time Difference of Arrival (TDOA). In two dimensions, for each pair of antennas, the emitter will be located somewhere on a hyperbola; so a third antenna on a different baseline will narrow the position down to a point. In three dimensions the hyperbolæ become surfaces of revolution about the line joining each pair, so four receivers will be needed. These four, or commonly still more receivers, will be set in a number of different baselines.

For long baselines (greater than a few tens of metres), this is a modification to the meaning of direction finding, in the sense that we are not locating the emitter on a ray described by one bearing. Rather, a hyperbola needs to be specified. The situation is simplified for baselines of perhaps ten or twenty metres ("short-baseline TDOA"), since the multiple bearings are then essentially the same, lying more or less on the hyperbola's asymptotes. This produces a straight line geometry and a simple direction. In this case the receivers can be sited on one platform, but the timing accuracy required then becomes crucial. The hyperbolic case becomes more important, and in principle easier to implement, when the receivers are more widely separated. But in practice, widely separated receivers are limited by the requirement that they both see the same pulse, and this is to some extent dictated by the emitter beamwidth. This topic is covered in some detail in Section 5.2, as well as in [1]. Typical bearing accuracy is 1–2°.

2.3 Interferometry

Whereas TDOA measures the difference in arrival times of a wave at two receivers, interferometry measures the difference in arrival *phase* at two receivers.

Beasley and Miles [2] has a good discussion on the various sources of error possible. Notable errors result from the following:

- Multipath interference due to the ESM platform's fuselage etc. This might produce bearings with significant errors, but if the receivers are changing orientation from one measurement to the next (e.g. if the platform is turning), then these errors will probably not lead to any bias in the measurements.
- Multipath interference caused by the radome, which is ever-present, but is at least a systematic error.
- Bearing ambiguity. This will not occur if the antenna spacing is less than half a wavelength. Typical radar wavelengths of 3–30 cm restrict the required antenna spacing, but in practice ambiguities are overcome by using multiple baselines.
- Thermal noise. This cannot easily be removed through system design, and hence may well be the greatest error contributor.

Interferometry tends to give very high accuracy: better than 0.5° is possible for single pulses on line-of-sight paths.

2.4 Absolute Doppler

When an emitter is at rest—for example anchored to Earth—its Doppler shift will give us information about where it is. This can be seen by realising that provided we know the emitter's velocity relative to us, together with its velocity radially from us, then we can in essence shift that velocity vector around our own position, until its radial component matches the measured value. All such matches can only lie on a cone extending out from ourselves, and this cone cuts Earth's surface in a hyperbola. So we'll know that the emitter lies somewhere on this hyperbola.

We *do* know the emitter's velocity relative to us because it is anchored to Earth, and so is just equal and opposite to our known velocity relative to Earth. We also know its radial velocity, since this is given by the ratio of observed to emitted frequencies:

$$\frac{f_{\mathrm{o}}}{f_{\mathrm{e}}} \simeq 1 - \frac{v_{\mathrm{radial}}}{c} \,. \tag{2.1}$$

The only real problem here is that we need to know the emitted frequency. And in practice, accurate knowledge of this is probably not available—especially because this frequency can change very quickly in modern radars. Thus, the method might only be of use for narrow-band signals produced by crystal-controlled radar, instead of the more common magnetron types whose frequencies are much less constant.

2.5 Frequency Difference of Arrival

Again for a stationary emitter, Frequency Difference of Arrival (FDOA) uses the difference in Doppler shifts in emitter frequency as observed by at least two platforms that both move. Since the Doppler shifts yield the radial velocity of each receiver from the emitter, it's a straightforward matter to determine the emitter location, which will be unique (or perhaps two-fold ambiguous) for most receiver configurations.

2.6 Differential Doppler

The Differential Doppler technique follows the same basic idea as FDOA. But whereas FDOA makes use of the same signal received by two receivers at widely separated points in space, in principle Differential Doppler needs just one receiver to measure a change in frequency over time, as the emitter moves relative to it. The complication arising is that the emitted frequency might change over time in an unknown way.

Beasley and Miles [2] discuss Differential Doppler, but actually have not properly incorporated the differential Doppler shift into their analysis. What they are calculating, the rate of increase of interferometer phase difference with time, has two parts: a pure Doppler component (i.e. involving $\mathrm{d}\lambda/\mathrm{d}t$) which they don't include in their analysis, and a sort of FDOA component caused by the motion of the emitter as seen by the receiver. This last term is the only one written down by Beasley and Miles, although it turns out to be the dominant one anyway. They doubt this technique has been tested. (A change is necessary to their equation (2-22): replace $\sin^2\phi$ with $\cos^2\phi$ since they have confused the two ϕ's in their Figures 2-1 and 2-8.)

Suitability for Different Bands. Narrow-band signals are best suited to Differential Doppler since they have a fairly well-defined wavelength. Also, their comparatively long pulse duration means that Differential Doppler is more effective than TDOA for this type

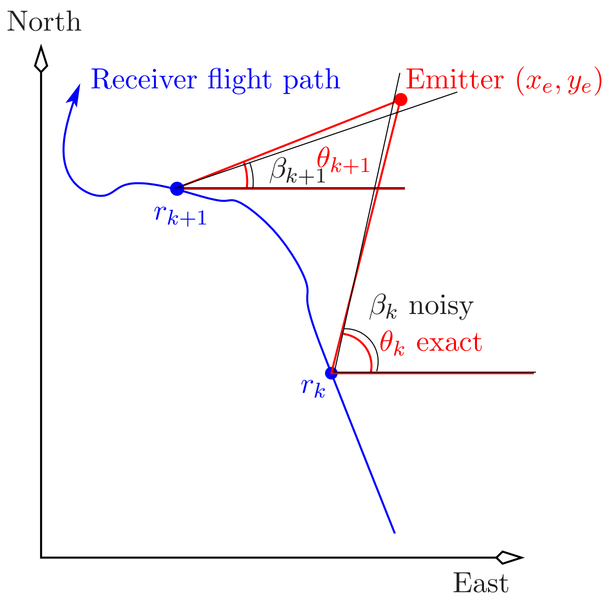

Figure 1: *Bearings taken at the k^{th} and $k{+}1^{th}$ receiver positions*

of signal. With wide-band signals the situation is reversed: the spread of frequencies means Differential Doppler does not perform well, so that TDOA is then preferred.

3 Statistical Processing of Bearing Data

3.1 Layout of a Geolocation Problem

For the next subsections, refer to the layout of Figure 1, adapted from [3]. A stationary emitter is situated at (x_e, y_e), while a receiver moving along some path takes bearing measurements at various points. The receivers are situated at $\boldsymbol{r}_1 \ldots \boldsymbol{r}_n$, and take (noisy) bearings $\beta_1 \ldots \beta_n$, which form the data set. What we wish to do is either produce an estimate of the emitter's position (x_e, y_e), or else, given such an estimate, improve upon it by producing a new estimate. We also require the error in the result. In general this can be difficult to calculate, but we can be guided as to what error can reasonably be expected by considering the Cramér–Rao lower bound for the given scenario, as well as the circular error probable (CEP) and range errors, as discussed next.

3.2 Cramér–Rao Lower Bound

The Cramér–Rao lower bound is an indicator of how well we can locate a parameter that is, in some sense, the "centre" of a distribution. Its use is motivated by this fact, since it will tell us how well a geolocation algorithm is performing compared to the absolute best possible performance theoretically attainable. This absolute benchmark also allows for a better comparison of the various algorithms, rather than just comparing them with each other.

Suppose we receive a signal from which we wish to extract an interesting parameter x. For example, in the case of a normal distribution, x might be the mean. In general, if a data set depends on this unknown parameter, then in the absence of our being able to determine x, we wish to define an “estimator” $\widehat{x}$ of x in such a way that $\widehat{x}$ is unbiased; by this is meant that calculating it from a collection of data sets would yield x on the average, so that the expected value $\mathbb{E}\{\widehat{x}\}$ equals x. In some situations the Cramér–Rao theory will produce this unbiased estimator for us, as well as a lower bound on the associated variance.

In an unpublished paper by Kim Brown at DSTO [4], this theory has been applied to the case of an aircraft flying at various broadside angles and ranges to an emitter, in an effort to calculate just what the minimum geolocation error will be, for the case of a physically realistic probability density function of bearing errors seen by the receiver. This function is chosen in [4] to be von Mises, a symmetrical bell-shaped curve with its maximum at zero degrees error, being the angular version of a gaussian distribution. (That is, it gives the probability density that the bearing suffers from some error, which can be positive or negative corresponding to the measured direction being too far off true in one direction or the other. The width of the bell curve is, as usual, related to the actual bearing error we attribute to the DF equipment.)

The results in [4] have been reproduced independently using Matlab in Appendix C. The Matlab programme goes further than [4], in the sense that it allows for arbitrary broadside angles, emitter ranges, receiver speeds, observation periods and number of standard deviations for the uncertainty ellipse, as opposed to the paper’s static values. The programme also does something a little different to the paper: it applies the von Mises probability distribution to the geolocation scenarios it is analysing (as opposed to the example in [4] which does not incorporate any tracking algorithms), and plots an uncertainty ellipse superimposed on the map with the tracking algorithm output.

This uncertainty ellipse gives us an idea of the best accuracy we can hope for when processing DF data, regardless of which method is being used to locate the emitter (always assuming a von Mises error, which is quite reasonable). The fact that the parameters can be freely input to Matlab means that we can use the Cramér–Rao theory as a basis for comparing the techniques of Cartesian Pseudo-Linear Estimator, Gauss–Newton, Recursive Least Squares, and Kalman filtering, as will be discussed shortly.

Cramér–Rao bounds are calculated for a multiple receiver scenario in Section 5.1. It must be added that the lower bound is, strictly speaking, only applicable to an unbiased estimator. But it turns out that the results of geolocation algorithms will generally tend to be biased, so that we must treat Cramér–Rao results cautiously.

3.2.1 Two Types of Cramér–Rao Bound

Signal processing theory involves two types of Cramér–Rao bound. These correspond to whether we use non-bayesian statistics or bayesian statistics for the geolocation calculation. In the sections that follow, we will describe several geolocation methods. The Cartesian Pseudo-Linear Estimator and Gauss–Newton are “batch” methods where the state of the emitter (position, velocity, etc.) is treated as having a definite but unknown value, with no prior knowledge assumed: this is a non-bayesian approach. One application of the batch method is sufficient to yield an estimate of that state. The last three methods are all "recursive", in which the state is treated as a random variable, according to some probability density function given initially by prior knowledge. This prior knowledge indicates that the Cramér–Rao calculation should assume a bayesian approach, since Bayes theory makes use of this prior knowledge.

The lower bound on the variance of the unbiased estimator can be shown to be the inverse of the *Fisher information* J (a matrix, in general). That is, if our unbiased estimator of the emitter's state is $\widehat{x}$, then Cramér–Rao theory states that the following is always true:

$$\operatorname{var}(\widehat{x}) \geqslant J^{-1} \equiv \text{CRLB, the Cramér–Rao lower bound,} \tag{3.1}$$

where the meaning of "greater than a matrix" is quantified just after (3.6) ahead. How do we calculate the Fisher information matrix? Consider first the non-bayesian case that is applied to batch processes.

Non-Bayesian Case. Suppose the data set of k measurements is

$$Z_k \equiv \{\boldsymbol{z}_1, \ldots \boldsymbol{z}_k\}\,, \tag{3.2}$$

which depends on an unknown parameter x (e.g., $\boldsymbol{z}_i$ are bearing/range pairs, and x is the position of the emitter). The set might also contain a first element $\boldsymbol{z}_0$, which is not a measurement as such, but rather is prior information. The probability that given some particular x, the k measurements Z_k will be observed is $p(Z_k|x)$, called the *likelihood.* It turns out to be more convenient to work with the negative logarithm of the likelihood:

$$L(Z_k|x) \equiv -\ln p(Z_k|x)\,. \tag{3.3}$$

For a one-dimensional case where x is a scalar, the non-bayesian Fisher information (a scalar) is defined as an expected value over the data:

$$J_{NB} \equiv \mathbb{E}_{Z_k}\left\{\frac{\partial^2 L(Z_k|x)}{\partial x^2}\right\} = \mathbb{E}_{Z_k}\left\{\left(\frac{\partial L}{\partial x}\right)^2\right\} \tag{3.4}$$

(where the last equality can be proved easily). Generally the data is dependent on several parameters, whose values at the latest time k are collectively called $\boldsymbol{x}_k$. Dropping the k subscripts, the Fisher information at time k is now a matrix:[1]

$$J_{NB} \equiv \mathbb{E}_Z\left\{\nabla_{\boldsymbol{x}}\left(\nabla^t_{\boldsymbol{x}} L(Z|\boldsymbol{x})\right)\right\} = \mathbb{E}_Z\left\{\left(\nabla^t_{\boldsymbol{x}} L\right)\left(\nabla_{\boldsymbol{x}} L\right)\right\}\,, \tag{3.5}$$

where ∇L is a row vector, and $\nabla^t L \equiv (\nabla L)^t$ is a column vector. Component-wise, the previous expression is:

$$J_{NB}\Big|_{i,j} \equiv \mathbb{E}_Z\left\{\frac{\partial^2 L(Z|\boldsymbol{x})}{\partial x_i\,\partial x_j}\right\} = \mathbb{E}_Z\left\{\frac{\partial L}{\partial x_i}\,\frac{\partial L}{\partial x_j}\right\} \tag{3.6}$$

(remembering that x_i, x_j are really the i, j components of $\boldsymbol{x}_k$). The inverse Fisher information J_{NB}^{-1} is a lower bound in the sense that $\operatorname{cov}(\widehat{\boldsymbol{x}}) - J_{NB}^{-1}$ is positive semidefinite—meaning that the quadratic form it produces is always $\geqslant 0$.

[1] In general, the CRLB theory assumes the *regularity condition* to hold: $\mathbb{E}_{Z_k}\{\nabla_{\boldsymbol{x}_k} L\} = 0$ for all $\boldsymbol{x}_k$, which is true for well behaved distributions.

Bayesian Case. The bayesian viewpoint used to calculate the Cramér–Rao bound, known as the "posterior" CRLB, for recursive routines is much more involved, and has been briefly described in Appendix A. It uses some recursive notation laid out in Section 3.10. Good references to its discussion of the posterior CRLB are [5, 6].

3.3 Circular Error Probable and Range Errors as Ways of Quantifying Error

There are various ways to quantify the accuracy of a geolocation method. Two in particular are used in this report. The first is the circular error probable (CEP). This is the radius of a circle that, when drawn around the actual position of an emitter, will encompass half of a large number of estimates to that position. So this circle comprises the region within which there is a 50% probability of estimating the emitter to lie; hence the smaller the CEP, the better is the geolocation method. Broadly speaking, since the area of the circle is proportional to the square of the CEP, a halving of the CEP really signals a dramatic improvement in accuracy. Although the collection of estimates made to the emitter position will not, in general, lie symmetrically around the true emitter position, nevertheless, the CEP is a good one-parameter quantifier of the accuracy of a geolocation method. In practice, the CEP will be drawn around the *estimated* position of the emitter, because the actual emitter position is unknown.

The second way to quantify the accuracy of a geolocation method is to specify two numbers: the cross- and down-range errors. Cross-range error is the transverse difference between the actual emitter position and our estimate of it. Down-range error is the difference between the range to the actual emitter and the range of the estimate, which is more or less as depicted in Figure 2 if the cross-range error is not too large. The figure shows a set of estimates to an emitter's position, with the CEP and range errors indicated.

3.4 Observability of the Emitter

In order to determine the receiver network necessary for a given geolocation problem—or perhaps the parameters of the flight path of one receiver—we need to give some consideration to whether an emitter's position can be uniquely determined, even in the absence of noise. Since our measurements are bearings only, we can imagine a scenario in which another, more agile, emitter moves arbitrarily, but always so as to remain hidden behind the real emitter. Thus the position of an emitter is never uniquely determined, since this other more agile emitter could really have been the only one present.

But this more agile emitter must have to follow a much more convoluted course than the visible one, and the wisest choice might be to say that whichever of these alternative emitters is moving in the simplest way is the real one. For example, suppose the receiver is moving at constant velocity: that is, a straight line with constant speed. Then an emitter that is really stationary will give rise to the same bearings as one that moves at just the right speed to always appear to be behind or in front of the the real one. And these could hide another one that is zig-zagging wildly back and forth, again always so as to remain

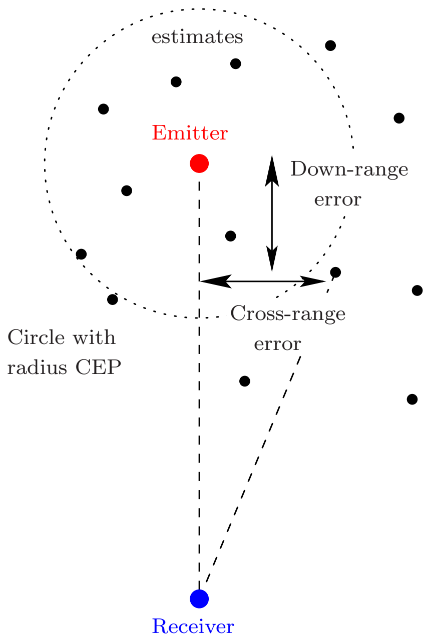

Figure 2: *Definitions of CEP and range errors*

hidden behind the real one. Our intuition tells us that those two that move are almost certainly not real, but rather are "ghosts". Depending on what prior information we have, the most pragmatic decision might be that the real emitter is in fact stationary.

Accepting the fact that the real emitter could be moving quite wildly in just the right way so as to give rise to the observed bearings, we denote the stationary emitter in the above example *observable* if, when we *model it as stationary*, a geolocation analysis will produce an accurate value for its position. Likewise a constant-velocity emitter is denoted as observable if, when it's modelled as having a constant velocity, a geolocation analysis returns an accurate value for its position and that constant velocity.

Relevant to this discussion is the idea of manoeuvring, which can be quantified in the following way. If two objects A and B are moving, then A manoeuvres more than B if, for example, A has a constant velocity while B only has a constant position; or A has a constant acceleration while B only has at most a constant velocity, and so on.

It can be shown that if the emitter is observed, then the receiver must have manoeuvred more to have found it. The converse is not quite true: just being more manoeuvrable does not guarantee that we will locate the emitter, but we usually will even so, provided we have made a reasonable model of its motion. The bottom line is that if we (the receiver) suspect the emitter is stationary, then we must at least be moving to geolocate it; while if we suspect the emitter has constant velocity then we must accelerate, even if that is limited to a single turn connecting two constant-velocity paths. While an acceptable acceleration might be nothing more than a kink in our otherwise constant-velocity flight path, the fact that this kink might be very slight shows that there is a continuum of degrees to which an emitter is observable. It is not true that emitters are either observable or not; the accuracy of a geolocation analysis depends on how we model the emitter's motion, and how we (the receiver) move. Different geolocation techniques have differing degrees of sensitivity to the emitter's observability.

All of this has the consequence of limiting what can be achieved, even in principle, with target motion analysis. When implementing a geolocation routine, we must specify the type of emitter motion: for example, if we suspect that the emitter has constant velocity, then we (the receiver) must accelerate to out-manoeuvre it in order for the geolocation to have any chance of making sense; and we also will need to provide some sort of model of the emitter that allows it to have a constant velocity.

It might be thought that since constant velocity is only a special kind of constant acceleration (where the acceleration is just zero), it might be useful to model the emitter as accelerating, even though we suspect it probably only has a constant velocity. Or, to take the situation to its logical conclusion, we might be able to model the emitter as having some high manoeuvrability, and so long as the receiver outmanoeuvres it in actuality, then it should be able to find the emitter's true motion. But this cannot be true. Since we can imagine that there is an infinite hierarchy of ever more complicated manoeuvring emitters hidden behind the real one, as soon as we model the emitter as anything of a high order motion, we will open ourselves up to confusing its motion with all of the ghost emitters that have manoeuvrability at most as complicated as that of our model. So there will be an ambiguity reflected in our results of where the emitter is.

In fact in practice, for some geolocation techniques ("smoothers" such as the CPLE technique described in Sect. 3.7), the more manoeuvrability we build into the model of the emitter, the closer its estimated position and motion lie to the *receiver's* position and motion. It actually appears to be sitting right on top of the receiver. With hindsight this is not as strange a result as it sounds, because such an emitter will certainly always be sighted in the direction given by the measured bearings. It's as if the geolocation routine is giving up, and simply returning an emitter state that will always "fit", in the sense of being compatible with the measured bearings.

3.5 Producing an Estimate of the Emitter Position: The Stansfield Algorithm

Perhaps the earliest of the "modern" papers to discuss the processing of bearing data was published by Stansfield in 1947 [7]. For the case where only noisy bearings are known of a single stationary emitter, the setup as described by Stansfield is loosely pictured in Figure 3.

Typically, the problem to be solved is that while the bearing lines due to two receivers intersect with no ambiguity, in general the lines of three receivers do not intersect. Instead, they form a so-called "cocked hat", a triangle within which the emitter might be thought probably to lie. In fact, in such a situation where the bearing line drawn is the centre of some probability distribution, there is only a 1 in 4 chance that the emitter will lie within the cocked hat triangle, despite the fact that somewhere within the triangle seems to be the best place to estimate its position in the absence of any other information. The best way out of this difficulty is to put any method of locating the emitter on a good statistical footing. We assume the bearings are spread according to some known probability distribution in bearing angle, peaked around the true bearing; and we require the most probable emitter position given this set of measured bearings. This is known as the *maximum a posteriori* estimate of the emitter's position. So we need to calculate the

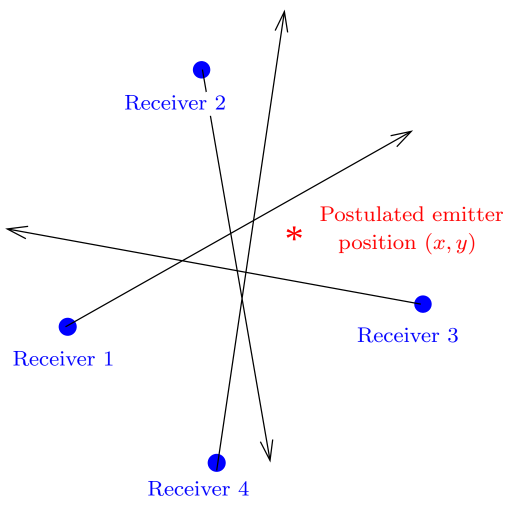

Figure 3: *Stansfield scenario*

probability that the emitter is at some position (x, y) given that bearing set, and vary x and y over the whole region where the emitter could possibly be, until we find a maximum.

It proves easier, but just as useful, to follow a slightly different approach known as the method of *maximum likelihood*. In this we take the true position of the emitter to be the one most likely to have given rise to the measured bearings. That is, we calculate the probability that the measured bearings will be what they are, for all possible emitter positions, and choose the position maximising this probability.

That is, the maximum a posteriori approach maximises

$$\text{prob}(\textit{emitter at } (x, y) \,|\, \textit{bearing data}) \,, \tag{3.7}$$

while the maximum likelihood approach maximises

$$\text{prob}(\textit{bearing data} \,|\, \textit{emitter at } (x, y)) \,. \tag{3.8}$$

These two probabilities are related via Bayes' rule.

Stansfield's original calculation is based around the following maximum likelihood analysis. Begin by referring to Figure 1 on page 5. The k^{th} measured bearing is just the exact bearing plus some noise:

$$\beta_k = \theta_k(x, y) + \nu_k \,. \tag{3.9}$$

Provided the bearing errors are normally distributed, the maximum likelihood probability will be a product of gaussian functions:

$$p(\beta_1 \dots \beta_n | \theta_1 \dots \theta_n) \propto \exp \sum_k \frac{-(\beta_k - \theta_k)^2}{2\sigma_k^2} \,. \tag{3.10}$$

Maximising this probability is equivalent to minimising twice the sum of squares in the exponent, this being known as the "cost function" of the problem:

$$\text{ML cost function} = \sum_k \frac{(\beta_k - \theta_k)^2}{\sigma_k^2} \,. \tag{3.11}$$

Stansfield expressed $\beta_k - \theta_k$ as a function of the emitter position (x, y), and then equated each of the x, y partial derivatives with zero. This can be done in different ways, depending on what level of accuracy is chosen and what assumptions are made. Different assumptions will lead to slightly different formulations and solutions of the problem.

While Stansfield's original paper used the maximum likelihood approach, it was written very obscurely. It was rewritten a decade later by Ancker [8] using a slightly more transparent approach. The calculation is complicated by the fact that the probability distributions are expressed fundamentally in terms of angles, so that when converted to distances using cartesian coordinates, the distance of the emitter from each receiver enters the problem in a mathematically complicated way. This is not a problem in itself; it just makes the resulting equations harder to solve.

Both Stansfield and Ancker tackled this difficulty by assuming that the distance from each receiver to the trial position of the emitter does not change as this trial position is varied. This approximation is viable, because the required probability will be nowhere near its maximum when this distance has begun to extend to regions where we know the emitter is certainly not present; thus we are really only considering the area around the peak of the probability—and this is precisely the only region in which we were interested anyway.

The so-called Stansfield solution for geolocation, as restated by Ancker, writes the emitter's best-estimate position in an expression that still involves these unknown distances, and as such is an iterative technique using a batch of bearings. Ancker suggested that initial estimates of these distances be taken from the centre of the largest polygon formed by the bearing lines. But calculating a good centre for the largest polygon formed becomes a problem in itself not unlike the original. Ancker does add that in the absence of such an estimate, we might resort to a trial and error approach.

In the following pages, we first consider two alternative formulations of Stansfield's problem, each having its own method of solution: the Cartesian Pseudo-Linear Estimator (CPLE), and the Gauss–Newton technique. These are "batch routines", in that they process a set of points to arrive at some estimate of the emitter position. This is followed by discussion and examples of a more modern, recursive, approach to geolocation, where only the latest bearing measurement is used to update the estimated emitter position. Sorenson [9] gives further background to the use of least squares in a DF context.

3.6 A Least Squares Primer

The discussions on the CPLE and Gauss–Newton routines that follow all use the formalism of the theory of least squares, which we review here. Suppose we have taken n measurements z_i, which in a noiseless world would fit linearly to a set of m parameters $x_1, \ldots, x_m$, using well-defined coefficients that comprise an $n \times m$ matrix H:

$$\begin{bmatrix} z_1 \\ \vdots \\ z_n \end{bmatrix} = H \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix} \tag{3.12}$$

or just

$$\boldsymbol{z} = H\boldsymbol{x}\,. \tag{3.13}$$

In general, we have an abundance of measurements: $n > m$ so that H is taller than square. In a noisy world the elements x_i are required to be found through the inexact relation

$$\boldsymbol{z} = H\boldsymbol{x} + \text{noise}\,, \tag{3.14}$$

where the noise cannot be known exactly. In general this set of equations is overdetermined, but we can find a best fit for $\boldsymbol{x}$ in a least-squares sense, by choosing $\boldsymbol{x}$ to minimise the distance between the two points in n-dimensional space, $H\boldsymbol{x}$ and $\boldsymbol{z}$. This is equivalent to choosing the $\boldsymbol{x}$ that minimises $|H\boldsymbol{x} - \boldsymbol{z}|^2$ (called a "cost function", which equals the squared length of the n-dimensional noise vector). This is done by setting the gradient of the cost function to zero:[2]

$$\begin{aligned} \nabla\left\{|H\boldsymbol{x} - \boldsymbol{z}|^2\right\} &= \nabla\left[(H\boldsymbol{x} - \boldsymbol{z})^t(H\boldsymbol{x} - \boldsymbol{z})\right] \\ &= \nabla\left[\boldsymbol{x}^t H^t H\boldsymbol{x} - 2\boldsymbol{z}^t H\boldsymbol{x} + \boldsymbol{z}^t\boldsymbol{z}\right] \\ &= 2\boldsymbol{x}^t H^t H - 2\boldsymbol{z}^t H\,. \end{aligned} \tag{3.16}$$

Equating the last expression with zero defines the least-squares solution $\bar{\boldsymbol{x}}$:

$$\bar{\boldsymbol{x}} = (H^t H)^{-1} H^t \boldsymbol{z} \equiv H^{\#}\boldsymbol{z}\,, \tag{3.17}$$

where $H^{\#}$ is the *pseudo inverse* of H. Numerical packages such as Matlab do not just calculate $H^{\#}$ by inverting and transposing as in (3.17). Numerical matrix inversion can be hazardous for near-singular matrices, so more sophisticated methods are used, such as singular value decomposition. Matlab's `pinv` function makes use of singular value decomposition (`pinv(H)` $= H^{\#}$).

Error Analysis

In practice, each measurement might be subject to some error σ_k. Assuming gaussian statistics, it is then more meaningful to minimise not the length of $H\boldsymbol{x} - \boldsymbol{z}$, but rather to weight each of its components by the corresponding inverse variance. That is, we should minimise

$$\sum_k \frac{(H\boldsymbol{x} - \boldsymbol{z})_k^2}{\sigma_k^2}\,. \tag{3.18}$$

This is identical to minimising a new cost function:

$$(H\boldsymbol{x} - \boldsymbol{z})^t P^{-1}(H\boldsymbol{x} - \boldsymbol{z}) \tag{3.19}$$

with respect to $\boldsymbol{x}$, where P is a diagonal matrix of variances:

$$P \equiv \begin{bmatrix} \sigma_1^2 & & 0 \\ & \ddots & \\ 0 & & \sigma_n^2 \end{bmatrix}. \tag{3.20}$$

[2] Use the following easily-verified results:

$$\nabla_{\boldsymbol{x}}\left(\boldsymbol{a}^t\boldsymbol{x}\right) = \boldsymbol{a}^t \quad \text{and} \quad \nabla_{\boldsymbol{x}}\left(\boldsymbol{x}^t A\boldsymbol{x}\right) = \boldsymbol{x}^t(A + A^t)\,. \tag{3.15}$$

The calculation is almost identical to that of (3.16), with the following result:

$$\bar{\boldsymbol{x}} = (H^t P^{-1} H)^{-1} H^t P^{-1} \boldsymbol{z}\,. \tag{3.21}$$

If all the errors are the same, i.e. $\sigma_1 = \cdots = \sigma_n$, then the variance matrix P cancels from this expression, leaving us with the usual pseudo inverse expression (3.17).

3.6.1 Total Least Squares: When the Errors are More Complicated

On a more advanced note, the theory of how best to manipulate matrices, when real-world noisy numbers are involved, can be used to improve upon the least-squares analysis of the preceding pages. The relevant theory is known as *singular value decomposition*, and deals with efficient ways to express matrices in terms of factors that are robust in numerical work. Robustness can be necessary when small numbers are involved in the calculations, that lie at the edge of the computing machine's inbuilt precision. Such numbers might be created in a numerical algorithm being used; matrix inversion is particularly prone to this. Geolocation using singular value decomposition is discussed in Appendix B.

3.7 Cartesian Pseudo-Linear Estimator Approach (CPLE)

Our first variant of the maximum likelihood approach (as is shown later) relies on the use of the orthogonality of vectors specifying bearings, as shown in Figure 4. In essence what the method does is locate a best point of intersection of the bearing lines with a least-squares fit. The calculation is simple enough to be done nonrecursively, and does not require an initial estimate of the emitter position. CPLE is in essence equivalent to the Stansfield algorithm, but the following description of it is conceptually simpler, and unlike the usual exposition of Stansfield, is easily extended to moving emitters.

The great utility of the CPLE is that although a batch process—a once-only processing of a given set of data—it is still accurate enough to compete with the other routines discussed ahead. It can be run on an initial data set and then used again on sets of subsequent data, but it can also be used to provide an initial estimate of where the emitter is, in order to supply one of the following algorithms with their required initial estimates of the emitter's position and state of motion.

In this section the CPLE method is outlined and the moving-emitter case is covered, in which the emitter is modelled as having at most a constant acceleration in two dimensions. Extension to three dimensions and higher derivatives of position is straightforward.

Suppose that the emitter position is a vector $\boldsymbol{s}(t)$. There are n receiver positions, where the k^{th} position is a vector $\boldsymbol{r}_k(t)$. Write all vectors as columns and work in cartesian coordinates. The emitter has initial position $\boldsymbol{s}_0$, initial velocity $\boldsymbol{v}_0$ and constant acceleration $\boldsymbol{a}$ (all vectors), so that

$$\boldsymbol{s}(t) = \boldsymbol{s}_0 + \boldsymbol{v}_0 t + \frac{1}{2}\boldsymbol{a}t^2\,. \tag{3.22}$$

Group the three constant vectors that we wish to find into one column vector $\boldsymbol{x}$. This has six elements if we are confined to the plane, since each of the vectors $\boldsymbol{s}_0, \boldsymbol{v}_0, \boldsymbol{a}$ then has

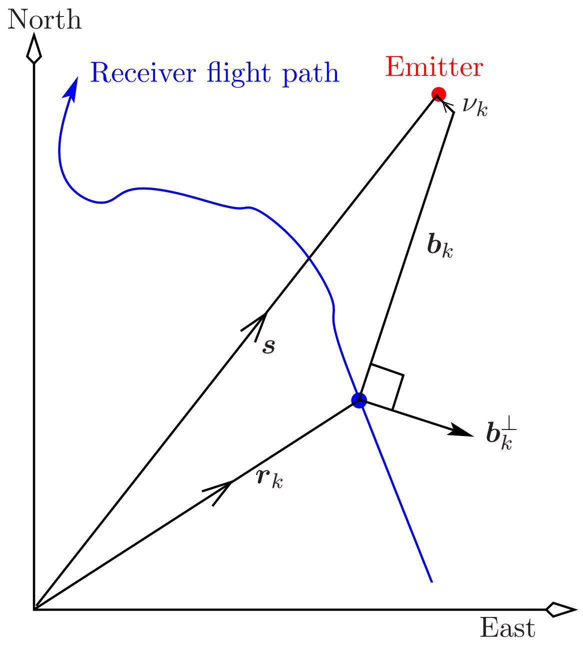

Figure 4: *Cartesian Pseudo-Linear Estimator approach*

two elements:

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{s}_0 \\ \boldsymbol{v}_0 \\ \boldsymbol{a} \end{bmatrix} . \tag{3.23}$$

In tandem with $\boldsymbol{x}$, introduce a matrix $A(t)$ that serves to absorb the emitter dynamics:

$$A(t) \equiv \begin{bmatrix} 1 & 0 & t & 0 & \frac{t^2}{2} & 0 \\ 0 & 1 & 0 & t & 0 & \frac{t^2}{2} \end{bmatrix} \quad \text{so that} \quad \boldsymbol{s}(t) = A(t)\, \boldsymbol{x} \, . \tag{3.24}$$

This leaves the state $\boldsymbol{x}$ of initial conditions remaining to be found.

The specifying of $\boldsymbol{x}$ has recast the problem into a stationary viewpoint. At some time t, receiver k makes a bearing measurement, and notes that the direction from it to the emitter is given by some not-quite-known noisy vector $\boldsymbol{b}_k(t)$, plus some noise $\boldsymbol{\nu}_k(t)$. (In practice the lengths of $\boldsymbol{b}_k(t), \boldsymbol{\nu}_k(t)$ are completely unknown or not even very well-defined, but that is of no consequence to the calculation.) We must estimate $\boldsymbol{x}$ given the set of all the $\boldsymbol{r}_k$ and $\boldsymbol{b}_k$. For each k we can write

$$\boldsymbol{s}(t) = \boldsymbol{r}_k(t) + \boldsymbol{b}_k(t) + \boldsymbol{\nu}_k(t) \, . \tag{3.25}$$

Now, every $\boldsymbol{b}_k$ has two unit vectors that are orthogonal to it. These turn out to be useful, although since we are working in two dimensions for simplicity, only one of these will be needed: call it $\boldsymbol{b}_k^\perp$, so that $\boldsymbol{b}_k^\perp \cdot \boldsymbol{b}_k = 0$. The utility of $\boldsymbol{b}_k^\perp$ is that although we have no knowledge of the length of $\boldsymbol{b}_k$, we *do* know its direction, which is enough to specify $\boldsymbol{b}_k^\perp$ exactly. That being the case, projecting all vectors along the $\boldsymbol{b}_k^\perp$ direction will then eliminate the unknown $\boldsymbol{b}_k$. This is effected by forming the dot product of both sides of (3.25) with $\boldsymbol{b}_k^\perp$, giving

$$\boldsymbol{b}_k^\perp \cdot \boldsymbol{s} = \boldsymbol{b}_k^\perp \cdot \boldsymbol{r}_k + \boldsymbol{b}_k^\perp \cdot \boldsymbol{\nu}_k \, . \tag{3.26}$$

If the k^{th} receiver takes its bearing measurement at time t_k (which times need not all be different), then the last equation becomes

$$\boldsymbol{b}_k^{\perp t} A(t_k)\boldsymbol{x} = \boldsymbol{b}_k^{\perp} \cdot \boldsymbol{r}_k + \boldsymbol{b}_k^{\perp} \cdot \boldsymbol{\nu}_k \,. \tag{3.27}$$

Writing this for each k gives

$$\begin{bmatrix} \boldsymbol{b}_1^{\perp} \cdot \boldsymbol{r}_1 \\ \vdots \\ \boldsymbol{b}_n^{\perp} \cdot \boldsymbol{r}_n \end{bmatrix} = \begin{bmatrix} \boldsymbol{b}_1^{\perp t} A(t_1) \\ \vdots \\ \boldsymbol{b}_n^{\perp t} A(t_n) \end{bmatrix} \boldsymbol{x} \;+\; \text{residual noise term.} \tag{3.28}$$

This will match the basic equation (3.14), provided we identify

$$H \equiv \begin{bmatrix} \boldsymbol{b}_1^{\perp t} A(t_1) \\ \vdots \\ \boldsymbol{b}_n^{\perp t} A(t_n) \end{bmatrix}, \quad z \equiv \begin{bmatrix} \boldsymbol{b}_1^{\perp} \cdot \boldsymbol{r}_1 \\ \vdots \\ \boldsymbol{b}_n^{\perp} \cdot \boldsymbol{r}_n \end{bmatrix}, \tag{3.29}$$

in which case the least-squares solution can be written down immediately:

$$\bar{\boldsymbol{x}} = H^{\#} z = \begin{bmatrix} \boldsymbol{b}_1^{\perp t} A(t_1) \\ \vdots \\ \boldsymbol{b}_n^{\perp t} A(t_n) \end{bmatrix}^{\#} \begin{bmatrix} \boldsymbol{b}_1^{\perp} \cdot \boldsymbol{r}_1 \\ \vdots \\ \boldsymbol{b}_n^{\perp} \cdot \boldsymbol{r}_n \end{bmatrix}. \tag{3.30}$$

Because the matrix H is built from noisy bearings, the method of total least squares might be useful here; so rather than apply (B10) or (3.30), we might wish to try (B11). In fact, it turns out that total least squares gives 10–20% smaller CEPs than the ordinary least-squares approach.

The CPLE is a simple locating method, yet gives very good results, even when using just the minimum number of data points required (for low noise). This number can be small. Consider, for example, the stationary emitter case in two dimensions, where just two bearings are needed (since these specify two lines with a well-defined intersection where the emitter can lie). Note that two bearing lines are also sufficient in three dimensions, because again we are only finding their intersection; since each line needs two numbers to specify it (altitude and azimuth), we have four numbers in total—more than enough information to solve for the three emitter coordinates. Alternatively, in three dimensions we are using *two* unit normals for each bearing line, so that with just two lines (3.26) is really four equations, and hence $\boldsymbol{s}$ is already overspecified.

Being a batch process, the CPLE routine can use all of the data to produce a one-off estimate of the emitter position, and incoming new bearings cannot be processed individually to refine this estimate. But in practice the CPLE gives a good estimate with very few data points, and so can always be run on a subset of data points that includes the latest one. Also, its simplicity means it can be used to generate an estimate quickly—which can then be used to seed one of the methods described in the next sections. Unlike the CPLE, these methods all require an initial estimate of the emitter state.

Error Analysis

Because the CPLE is a straightforward application of the least-squares technique, the way to calculate the error in the estimated emitter position (x_e, y_e) is known from standard statistical theory. If the bearing error is a constant σ, then the relevant covariance matrix is [3, 10]:

$$\text{cov}(x_e, y_e) = \sigma^2 (H^t H)^{-1}. \tag{3.31}$$

As discussed in Appendix C, if we calculate this matrix and use it to draw an error ellipse centred on the estimate point, the result is usually correlated well with the actual performance of the algorithm in terms of where it estimates the emitter to be on repeated runs. This error ellipse also tends to correlate well with the Cramér–Rao lower bound ellipse—but not always. It might be much larger than the Cramér–Rao ellipse but can also be somewhat smaller. It *does* tend to have about the same size and orientation. Presumably the departure from Cramér–Rao is caused by a bias in the CPLE.

3.7.1 Relating the CPLE to the Method of Maximum Likelihood

The least-squares solution to the CPLE turns out to be a simplified form of the method of Maximum Likelihood. We can show this by comparing the CPLE cost function to the maximum likelihood cost function. Note from (3.27, 3.28, 3.29) that $|H\boldsymbol{x} - \boldsymbol{z}|^2$ is the sum of squares of the noise terms $\boldsymbol{b}_k^\perp \cdot \boldsymbol{\nu}_k$. Writing d_k as the true distance from receiver k to the possibly-moving emitter at time t_k, this sum of squares becomes a sum of squares of perpendicular distances:

$$\begin{aligned} \text{CPLE cost function} \quad &\equiv \quad |H\boldsymbol{x} - \boldsymbol{z}|^2 = \sum_k \left(\boldsymbol{b}_k^\perp \cdot \boldsymbol{\nu}_k\right)^2 \\ &= \quad \sum_k d_k^2 \sin^2(\beta_k - \theta_k) \\ &\simeq \quad \sum_k d_k^2 (\beta_k - \theta_k)^2 \, . \end{aligned} \tag{3.32}$$

In contrast, the maximum likelihood cost function was calculated on page 11 to be:

$$\text{ML cost function} = \sum_k \frac{(\beta_k - \theta_k)^2}{\sigma_k^2} \, . \tag{3.33}$$

This compares well with the CPLE cost function, provided that (a) our lack of knowledge of the distances d_k means we are prepared to set all of these to be constant, so that they factor out of (3.32); and (b) the bearing errors are all the same, so can be factored out of (3.33). The CPLE does indeed stand on a very solid statistical footing.

3.8 Gauss–Newton/MLE Algorithm

Given a set of bearings, and wishing to use a least-squares fit to the set of parameters required, a very straightforward and widely-known approach again applies the maximum likelihood idea, but this time to the angular difference between true and observed emitter positions. It has attracted the name *maximum likelihood estimation*, and the usual solution method used is the Gauss–Newton method of following a gradient to locate a minimum point. These are generic names, but both have been attached to this particular method, which is termed the Gauss–Newton algorithm in this report and elsewhere. Further discussion can be found in the reference by Foy [11]. There, Foy compares the Gauss–Newton and Kalman routines (described later), noting that Gauss–Newton is more desirable due to its simplicity. However, like the CPLE, the Gauss–Newton algorithm is a batch technique, and so fits into a different class of routines than does the Kalman filter.

Begin by relating measured to exact bearings:

$$\beta_k = \theta_k(\boldsymbol{x}) + \nu_k \,. \tag{3.34}$$

We wish to recast this equation in the matrix language of Section 3.6, so as to solve it in a least-squares sense. One way to do this first estimates the emitter state (3.23) to be some $\boldsymbol{x}_0$, and then Taylor expands (3.34) around this estimate to first order:

$$\beta_k \quad \simeq \quad \theta_k(\boldsymbol{x}_0) + \nabla\theta_k(\boldsymbol{x}_0)(\boldsymbol{x} - \boldsymbol{x}_0) + \nu_k \,, \tag{3.35}$$

where the gradient is

$$\nabla\theta_k = \begin{bmatrix} \frac{\partial\theta_k}{\partial(s_{0x})} & \frac{\partial\theta_k}{\partial(s_{0y})} & \frac{\partial\theta_k}{\partial(v_{0x})} & \frac{\partial\theta_k}{\partial(v_{0y})} & \frac{\partial\theta_k}{\partial(a_{0x})} & \frac{\partial\theta_k}{\partial(a_{0y})} \end{bmatrix} . \tag{3.36}$$

With n measurements taken, write (3.35) as

$$\begin{bmatrix} \beta_1 - \theta_1(\boldsymbol{x}_0) \\ \vdots \\ \beta_n - \theta_n(\boldsymbol{x}_0) \end{bmatrix} \simeq \begin{bmatrix} \nabla\theta_1(\boldsymbol{x}_0) \\ \vdots \\ \nabla\theta_n(\boldsymbol{x}_0) \end{bmatrix} (\boldsymbol{x} - \boldsymbol{x}_0) + \boldsymbol{\nu} \,, \tag{3.37}$$

to be solved for $\boldsymbol{x}$. Writing

$$H \equiv \begin{bmatrix} \nabla\theta_1(\boldsymbol{x}_0) \\ \vdots \\ \nabla\theta_n(\boldsymbol{x}_0) \end{bmatrix} , \quad z \equiv \begin{bmatrix} \beta_1 - \theta_1(\boldsymbol{x}_0) \\ \vdots \\ \beta_n - \theta_n(\boldsymbol{x}_0) \end{bmatrix} , \tag{3.38}$$

we see that this equation now matches (3.14), so that we can immediately write down an updated state estimate $\boldsymbol{x}_1$ with a least-squares approach:

$$\boldsymbol{x}_1 = \boldsymbol{x}_0 + \begin{bmatrix} \nabla\theta_1(\boldsymbol{x}_0) \\ \vdots \\ \nabla\theta_n(\boldsymbol{x}_0) \end{bmatrix}^{\#} \begin{bmatrix} \beta_1 - \theta_1(\boldsymbol{x}_0) \\ \vdots \\ \beta_n - \theta_n(\boldsymbol{x}_0) \end{bmatrix} . \tag{3.39}$$

This, the Gauss–Newton algorithm, processes a batch of measurements and iterates to converge (hopefully) to the actual emitter state. Simulations show that the method of total least squares does not work anywhere near as well for the Gauss–Newton algorithm as does normal least squares.

Use the notation from the previous pages, and set $\boldsymbol{s}_k$ to be the emitter position at the time of measurement k. (That is, we are free to model the emitter as having e.g. constant velocity, constant acceleration, and so on.) If we set

$$\begin{bmatrix} \Delta x_k \\ \Delta y_k \end{bmatrix} \equiv \boldsymbol{s}_k - \boldsymbol{r}_k \,, \tag{3.40}$$

then the true bearings are given by

$$\theta_k = \tan^{-1} \frac{\Delta y_k}{\Delta x_k} + \text{quadrant-dependent constant.} \tag{3.41}$$

In that case, if the emitter motion is modelled by e.g. constant acceleration, then the gradient with respect to $s_{0x}, \ldots, a_y$ can be written down:

$$\nabla\theta_k = \frac{1}{\Delta x_k^2 + \Delta y_k^2} \begin{bmatrix} -\Delta y_k & \Delta x_k \end{bmatrix} \begin{bmatrix} 1_{2\times2} & t_k \, 1_{2\times2} & t_k^2/2 \, 1_{2\times2} \end{bmatrix},$$

$$\text{where} \quad 1_{2\times2} \equiv \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}. \tag{3.42}$$

The new estimate $\boldsymbol{x}_1$ will lead to new values of z and H, so that we must iterate until hopefully the estimates converge to within some tolerance—although there is no guarantee that this limit will be the actual emitter position. (Again, a bias is present in this technique, just as occurs in the CPLE.) In practice, we step between iterations by an amount larger than that given in (3.39): that is, we introduce a gain, being some number whose size is found by experiment (but is roughly 1), and multiply this gain by the step in (3.39). In general, increasing the gain speeds up convergence—but if the steps are too large then the algorithm will become oscillatory or unstable.

In writing code for these routines, we need to be aware that the lone inverse tangent function needs to be supplemented with a constant in order to return angles within the full 360° range. Of course, the constant differentiates to zero so need not be explicitly written here—although it certainly must be inserted into code that implements the routine. But even this is not enough. In the actual processing, we are ultimately comparing measured bearings with theoretical values. Depending on the convention chosen, there must always be a critical emitter position where the theoretical value jumps by 360° as the emitter is moved slightly. This lack of continuity will lead to instability in the routine, unless the code is written carefully to take account of this fact.

Error Analysis

Although some sort of error analysis can be introduced into the one-run CPLE, a similar approach to the iterative Gauss–Newton algorithm is not clear-cut. Each iteration is a simple least-squares process; however, what results is not an estimate of the emitter position, but rather an estimate of the step size needed to move away from the initial estimate, together with the error in this step size, should we wish to calculate this error by the same approach as was used for the CPLE. But we cannot just start out with some initial estimate with its own error, and then simply add more errors as several more steps are taken, since the errors are presumably not independent: this approach would yield a final error much greater than what is actually observed through running simulations. Gauss–Newton can give quite accurate results even with a very inaccurate initial estimate, but by no means always—it can also diverge wildly when the other methods described here are well-behaved.

Spingarn [3] writes the error for Gauss–Newton as our (3.31). For scenarios in which Gauss–Newton does converge, the final error is described fairly well by the ellipse plotted from the covariance matrix of (3.31). This error ellipse also usually approximately matches the Cramér–Rao bounding ellipse. An alternative representative error for Gauss–Newton is often found by computing the CRLB at the final estimate of the state $\boldsymbol{x}$.

Note that like the CPLE, Gauss–Newton is also a batch routine, in that it requires a new observation to be added to at least some of the previous ones before running the algorithm. In this sense it might be considered wasteful, since at least some old measurements need to be reprocessed. The need for a streamlined approach led historically to the Recursive Least Squares class of algorithms.

3.9 Recursive Least Squares (RLS)

Recursive Least Squares algorithms form a class of update algorithms that process the latest data point only. Such techniques have been known for the last half century, and two of the more widely known ones are considered here. The first is a procedure that dates back to 1950 if not earlier [12], and this, at least in the form used by Godard in 1974, has come to be called simply the Recursive Least Squares algorithm, and is very popular [13]. The second algorithm is the famous Kalman filter, due to Kalman in 1960 [14].

The principle behind these routines is that since we have already made use of all except the latest data point, any step in the algorithm that uses all of the data should be re-expressible in terms of past values and the latest data. This results in an update, using only the latest data, of the last state estimate. Because of this, the amount of data needing to be manipulated is kept at a constant small size: e.g. the matrix H of the previous two routines is now always composed of just one row, without growing a new row with each incoming data point, as it would do in the previous routines.

The usual RLS routine as presented here is derived in [15], although it is here extended to the constant acceleration case for the emitter. As usual, the extension to higher order motions is straightforward. The matrices H and z of the previous sections are now replaced by the latest measurements only, giving them sizes 1×6 and 1×1 respectively. In analogy to (3.38) we write

$$H = \nabla\theta(\boldsymbol{x}) \,, \quad z = \beta - \theta(\boldsymbol{x}) \,. \tag{3.43}$$

The procedure followed using RLS is, however, more complicated than that of Gauss–Newton. It's written in the following way so as to be suggestive of the Kalman filter, discussed next:

1. Start with some estimate of the matrix of variances P [as in (3.20)], equal to some large number b times the 6×6 identity. A good choice of b is crucial and needs to be determined empirically.

2. Choose a number r between 0 and 1, again to be determined empirically. This defines a sort of fading memory, in the sense that it allows more weight to be given to more recent measurements.

3. Calculate H, z from (3.43).

4. Calculate a gain $K = PH^t/(r + HPH^t)$.

5. Now update the emitter position:

$$\boldsymbol{x} \longrightarrow \boldsymbol{x} + Kz\,. \tag{3.44}$$

6. Finally update P, and then return to step 3:

$$P \to (1_{6\times 6} - KH)P/r\,. \tag{3.45}$$

This algorithm enables us to keep updating the latest estimate of the emitter position, by only ever just incorporating the latest measurement as it arrives.

3.10 Kalman Filter

The Recursive Least Squares algorithm is not always stable, and several attempts have been made to improve on it. The best known is due to Kalman [14], and for linear gaussian models the Kalman filter is the only one in wide use [16]. Its basic principles are explained well by Sorenson [9], whose notation is used here. The algorithm has been rederived recently in an accessible way by Challa and Koks [17], using the ideas of Bayes theory. The algorithm is conventionally called the "Kalman filter", because, like Recursive Least Squares, it *filters* the data to give a predicted estimate of the state at each timestep. This differs from methods such as CPLE and Gauss–Newton, which are *smoothers*, in that they operate on a set of data to produce just one state, which is then used to predict the future motion of the emitter.

A basic premise underlying Kalman theory is that the system being observed evolves in some way able to be modelled. In the case of tracking, the emitter might be modelled with e.g. constant velocity, so that its state comprises its position and velocity. In general its state at the time of the k^{th} measurement can be written as some vector $\boldsymbol{x}_k$ whose evolution we model as follows:

$$\boldsymbol{x}_{k+1} = F_k \boldsymbol{x}_k + G_k \boldsymbol{v}_k + \boldsymbol{u}_k\,. \tag{3.46}$$

Here, F_k is some evolution matrix allowing us to specify a model of how the emitter's state changes. $\boldsymbol{v}_k$ is the "process" noise, a term that models how the emitter's motion might depart from what our model assumes through the F_k term. For the purpose of investigation we have built the filter by assuming the emitter moves in some complicated way, by putting the effects of acceleration (and its derivatives, if need be) into the evolution matrix F_k, as is done in [18, 19]. We can also model any jitter in the target by another kind of acceleration term, but rather than explicitly putting it into F_k, we include it in a stochastic way through the matrix G_k that multiplies the noise. Finally, $\boldsymbol{u}_k$ is any extra non-noise term required, such as a shift in coordinates.

The state $\boldsymbol{x}_k$ is not observed directly; rather, we make a measurement z_k:

$$z_k = H_k \boldsymbol{x}_k + w_k + y_k\,. \tag{3.47}$$

Here, H_k describes the measurement process in terms of the system state, w_k is any noise introduced by the measurement, and y_k is any extra term required as part of the model. The process and measurement noises are assumed white and gaussian:

$$\boldsymbol{v}_k \sim N(0, Q_k)\,, \quad w_k \sim N(0, R_k)\,. \tag{3.48}$$

The standard terminology also defines:

$$\begin{aligned}\widehat{\boldsymbol{x}}_{k|k-1} &\equiv \text{predicted value ("predictor") of } \boldsymbol{x}_k \text{ determined after measurement } k-1\\ \widehat{\boldsymbol{x}}_{k|k} &\equiv \text{best estimate ("filtered estimate") of } \boldsymbol{x}_k \text{ determined after measurement } k.\end{aligned} \tag{3.49}$$

The Kalman filter also allows us to predict the error in $\boldsymbol{x}_{k|k-1}$ and $\boldsymbol{x}_{k|k}$ via the covariance matrices:

$$\begin{aligned}P_{k|k-1} &\equiv \mathbb{E}\{(\boldsymbol{x}_k - \widehat{\boldsymbol{x}}_{k|k-1})(\text{same})^t\} = \text{covariance of error in predicted estimate of } \boldsymbol{x}_k\\ P_{k|k} &\equiv \mathbb{E}\{(\boldsymbol{x}_k - \widehat{\boldsymbol{x}}_{k|k})(\text{same})^t\} = \text{covariance of error in filtered estimate of } \boldsymbol{x}_k.\end{aligned} \tag{3.50}$$

The Kalman filter is implemented through the following procedure:

1. Estimate $\widehat{\boldsymbol{x}}_{0|0}, P_{0|0}, Q_k, R_k\ \forall k$.

   For $k = 1$ onwards, calculate the following:

2. $P_{k|k-1} = F_{k-1}\, P_{k-1|k-1}\, F^t_{k-1} \;+\; G_{k-1}\, Q_{k-1}\, G^t_{k-1}$
3. $\widehat{\boldsymbol{x}}_{k|k-1} = F_{k-1}\, \widehat{\boldsymbol{x}}_{k-1|k-1} + \boldsymbol{u}_{k-1}$
4. $K_k = P_{k|k-1}\, H^t_k\, \left(H_k\, P_{k|k-1}\, H^t_k + R_k\right)^{-1}$
5. $\widehat{\boldsymbol{x}}_{k|k} = \widehat{\boldsymbol{x}}_{k|k-1} + K_k\, \left(z_k - H_k\, \widehat{\boldsymbol{x}}_{k|k-1} - y_k\right)$
6. $P_{k|k} = (I - K_k\, H_k)\, P_{k|k-1}$ (3.51)

Note that $P_{k|k-1}, K_k$ and $P_{k|k}$ can be calculated offline if the process and measurement matrices are known for all time, as they often are; and this makes the linear Kalman filter very fast to implement.

The Kalman filter does not have the same sort of gain as found in Gauss–Newton, nor a fading memory parameter as found in RLS. It simply combines the state inferred from the latest measurement, with the current state of the system as predicted from the other measurements to date, using appropriate weightings; and it does this in a statistically robust and optimal way.

Despite its many components, there is no common notation in general use for the Kalman filter. For example, the noise vector and scalar $\boldsymbol{v}$ and w are commonly called $\boldsymbol{w}$ and v.

Extended Kalman Filter (EKF)

The linear Kalman filter gives the best possible fit for linear problems with gaussian noise, i.e. where the state parameters and the measurements taken at time $k+1$ are linear combinations of the state parameters at time k. But no such claim can be made for the nonlinear case, such as is encountered in geolocation through (3.41). The modification made to the linear filter to incorporate nonlinearity gives rise to the *Extended* Kalman filter (EKF). We consider here two types: the first uses cartesian coordinates, while the second, both more effective and more complex, uses a modified set of polar coordinates.

To extend the linear filter's application to these nonlinear cases, start with analogues of the linear equations (3.46, 3.47):

$$\begin{aligned} \boldsymbol{x}_{k+1} &= f_k(\boldsymbol{x}_k) + g_k(\boldsymbol{v}_k) \\ z_k &= h_k(\boldsymbol{x}_k) + w_k\,, \end{aligned} \tag{3.52}$$

and linearise these assuming $\boldsymbol{x}_k - \widehat{\boldsymbol{x}}_{k|k}$ to be small, using expressions like

$$\begin{aligned} f_k(\boldsymbol{x}_k) &\simeq f_k(\widehat{\boldsymbol{x}}_{k|k}) + F_k \cdot (\boldsymbol{x}_k - \widehat{\boldsymbol{x}}_{k|k}) \\ &= F_k\,\boldsymbol{x}_k + f_k(\widehat{\boldsymbol{x}}_{k|k}) - F_k\,\widehat{\boldsymbol{x}}_{k|k}\,. \end{aligned} \tag{3.53}$$

F_k is a matrix whose i^{th} row is the gradient of the i^{th} component of f_k, with respect to the set of state parameters. Similar equations hold for g_k and h_k and define G_k, H_k. (It seems to be normal to expand both f_k and g_k around $\widehat{\boldsymbol{x}}_{k|k}$ but h_k around $\widehat{\boldsymbol{x}}_{k|k-1}$; this is just an arbitrary convention, but we have followed suit.) The filter is applied in the following way:

1. Estimate $\widehat{\boldsymbol{x}}_{0|0}, P_{0|0}, Q_k, R_k\ \forall k$ as before.

   Then for $k = 1$ onwards:

2. $P_{k|k-1} = F_{k-1}\,P_{k-1|k-1}\,F_{k-1}^t \ +\ G_{k-1}\,Q_{k-1}\,G_{k-1}^t$
3. $\widehat{\boldsymbol{x}}_{k|k-1} = f_{k-1}(\widehat{\boldsymbol{x}}_{k-1|k-1})$
4. $K_k = P_{k|k-1}\,H_k^t\left(H_k\,P_{k|k-1}\,H_k^t + R_k\right)^{-1}$
5. $\widehat{\boldsymbol{x}}_{k|k} = \widehat{\boldsymbol{x}}_{k|k-1} + K_k\left[z_k - h_k(\widehat{\boldsymbol{x}}_{k|k-1})\right]$
6. $P_{k|k} = (I - K_k H_k)\,P_{k|k-1}$ (3.54)

Two problems arise in a nonlinear scenario. The first is that since we are always expanding around the latest estimate of the state vector, the matrices $P_{k|k-1}, K_k$, and $P_{k|k}$ can no longer be calculated offline, and this adds greatly to the time taken to run the filter. The second problem is that since the covariance matrix $P_{k|k}$ depends on H_k—which is now being calculated from the latest estimates of the state vector in a similar way to (3.53)—the covariances inherit the inaccuracy of this state vector, and so can no longer be relied upon to provide the true errors in the results. Hence there appears to be no good method of estimating the accuracy of the result, apart from measuring the spread of estimated emitter positions or tracks based on a Monte Carlo approach. Worse, the inaccuracies in $P_{k|k}$ now feed back into the Kalman equations and can produce instabilities.

Cartesian EKF

The tracking problem can be formulated in the following way for the case of an emitter with constant acceleration in cartesian coordinates. Remember that in the RLS and Kalman approaches, we don't characterise the state of the emitter by the constants parametrising its motion in the way that we did for the CPLE and Gauss–Newton. Rather, the emitter's state vector $\boldsymbol{x}_k$ reflects its state at the instant k. If we write its position at k as (x_k, y_k) (hoping that the use of both $\boldsymbol{x}_k$ and x_k does not cause confusion!), then the state vector is (with dots denoting time rates of change, and a transpose used to save vertical space):

$$\boldsymbol{x}_k = \begin{bmatrix} x_k & y_k & \dot{x}_k & \dot{y}_k & \ddot{x}_k & \ddot{y}_k \end{bmatrix}^t . \tag{3.55}$$

Next we write the state equation

$$\boldsymbol{x}_{k+1} = \begin{bmatrix} 1 & 0 & \Delta t & 0 & \Delta t^2/2 & 0 \\ 0 & 1 & 0 & \Delta t & 0 & \Delta t^2/2 \\ 0 & 0 & 1 & 0 & \Delta t & 0 \\ 0 & 0 & 0 & 1 & 0 & \Delta t \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \boldsymbol{x}_k \equiv F_k \boldsymbol{x}_k . \tag{3.56}$$

Writing

$$\begin{bmatrix} \Delta x_k \\ \Delta y_k \end{bmatrix} \equiv \begin{bmatrix} x_k \\ y_k \end{bmatrix} - \boldsymbol{r}_k , \tag{3.57}$$

then the measurement model is just the bearing seen by the receiver:

$$z_k = \underbrace{\tan^{-1} \frac{\Delta y_k}{\Delta x_k}}_{\equiv \text{ Kalman } h_k(\boldsymbol{x}_k)} + \text{noise}, \tag{3.58}$$

in which case we have

$$h_k(\boldsymbol{x}_k) \simeq h_k(\widehat{\boldsymbol{x}}_{k|k-1}) + \underbrace{\begin{bmatrix} \frac{\partial h_k}{\partial x_k} & \frac{\partial h_k}{\partial y_k} & \frac{\partial h_k}{\partial \dot{x}_k} & \frac{\partial h_k}{\partial \dot{y}_k} & \frac{\partial h_k}{\partial \ddot{x}_k} & \frac{\partial h_k}{\partial \ddot{y}_k} \end{bmatrix}_{\widehat{\boldsymbol{x}}_{k|k-1}}}_{\equiv \text{ Kalman } H_k} (\boldsymbol{x}_k - \widehat{\boldsymbol{x}}_{k|k-1}) , \tag{3.59}$$

where

$$H_k = \frac{1}{\Delta x_k^2 + \Delta y_k^2} \begin{bmatrix} -\Delta y_k & \Delta x_k & 0 & 0 & 0 & 0 \end{bmatrix} . \tag{3.60}$$

The algorithm can now be implemented, and this has been done using Matlab as described ahead in Appendix C.

Challa and Faruqi [20] have made a comparison of two approaches to using this filter for a stationary emitter. They use two versions of the state vector $\boldsymbol{x}_k$. The first is the one used here, i.e. the emitter position, a 2×1 vector. The other version they use is a 4×1 vector composed of the emitter's position and its velocity relative to the receiver, with all components expressed as fractions of the range. They then show, by running several simulations of Spingarn's example [3], that the second choice of state vector leads to better results: faster convergence, and also more accurate convergence in the presence of noise. This begs the question of what actually *is* the best choice of state vector, and this is a large area of ongoing research in the geolocation community, that has not been investigated in this report.

Error Analysis

Like Gauss–Newton, errors in the Recursive Least Squares and Kalman routines are also difficult, if not impossible, to evaluate. When the Kalman filter is applied to linear tasks, the errors in its state estimates are given by the entries of the covariance matrix $P_{k|k}$; but for a linear approximation to the nonlinear problem of geolocation as used here, the $P_{k|k}$ matrix loses some of its covariance meaning even though its role in the Kalman algorithm is unchanged. See the discussion on page 23.

Since both RLS and the Kalman filter are iterated as each new data point comes in, the covariance evolves over time. Thus, while e.g. the error in the x component of position as seen in $P_{k|k}$ might get smaller to an acceptable final value, the y position error may well start out by getting smaller, but might soon begin to grow; it might even become much larger than the final error in the estimated placement of the emitter. Or, it might shrink to zero. As in the Gauss–Newton case and to some extent the CPLE also, it could well be that the best way to calculate a state estimate error is through a Monte Carlo approach, by processing repeated runs of the receiver flight path and looking at the scattering of estimates to the emitter position that result.

Modified Polar Coordinate EKF

The Cartesian EKF is not always stable and can give biased estimates, as shown explicitly in [20]. Variations have been proposed that are based instead on polar coordinates [20, 21]. We consider one such well known way here: the Modified Polar coordinate EKF algorithm (MPEKF) [19, 22, 21]. The increased stability provided by this algorithm is paid for initially by a more complicated choice of state vector. In the notation of Arulampalam [19], if β_k is the bearing angle of the emitter as seen by the receiver measured clockwise from north (i.e. Arulampalam's $\beta_k = \pi/2 -$ our β_k in Figure 1), and if Arulampalam's r_k is the range to the emitter measured from the receiver, then the state vector corresponding to (3.55) is set to be

$$\boldsymbol{x}_k = \begin{bmatrix} \dot{\beta}_k & \dfrac{\dot{r}_k}{r_k} & \beta_k & \dfrac{1}{r_k} \end{bmatrix}^t . \tag{3.61}$$

Unlike the previous state vectors described, this is independent of whether or not the emitter accelerates, since we are now modelling the effects of acceleration only by including a nonzero matrix Q_k, the covariance of the process noise. So this form for the state vector $\boldsymbol{x}_k$ suffices no matter how the emitter's motion is modelled. The more complex entries of the state vector mean that the implementation of the filter is more involved: refer to [19] for the details of calculating the initial matrices that model covariances and how the system evolves.

As shown in Appendix C, the performance of the MPEKF is dependent on a good initialisation, which of course is something that cannot necessarily be achieved in practice. The filter's performance has also been analysed in [21], and is shown there to be stable and asymptotically unbiased.

4 Review of Selected Papers in the Field

Stansfield's original geolocation paper [7] became much cited in the literature, although the main effect it had was to spark interest in finding efficient ways to solve the geolocation problem. At the time, nomograms and specially graded least-squares rulers were used for working with charts overdrawn with bearing lines, but the advent of more computing power also saw the rise of more complex methods, such as the Kalman filter, that depend for their use on computers.

The Kalman filter is often seen as fundamental to tracking (and for example is used in Spingarn's widely-quoted paper [3]), but the simpler Gauss–Newton method is still quite comparable in its performance. This is especially true with more powerful computers that allow the fast reprocessing of data as per Gauss–Newton's needs. For example, Poirot and M^{c}Williams [23] apply Gauss–Newton successfully to bearings as measured on Earth (i.e. using latitude and longitude). Rao and Reddy [24] also deal with Earth using a least-squares approach that makes use of new combinations of the various bearing parameters, and by combining this new routine with a Kalman filter, have produced a more efficient algorithm. This highlights an important point in the search for more efficient algorithms: by solving the problem using new and unusual combinations of the conventionally used variables, an increase in efficiency can sometimes be obtained (as is done in [20]); but an approach to *producing* these unusual combinations is not clear.

Elliott and others [16] have recently produced an algorithm that generalises the Kalman filter and appears to perform much better, but their analysis is somewhat obscure, and the algorithm is not considered in this report.

Gavish and Weiss [25] compare the performance of a least-squares algorithm with the more involved Stansfield algorithm. They state a well-known theorem that says the Maximum Likelihood Estimator is unbiased, and can achieve the Cramér–Rao lower bound in its accuracy provided the number of measurements is large enough. By running scenarios using both a least-squares approach and the Stansfield algorithm, they find that the least-squares bias vanishes as the number of measurements increases, while the Stansfield results are biased and this bias does not vanish, and can even grow, as the number of measurements increases.

Mahapatra takes a different approach to geolocation in general [26]. He suggests flying a receiver flight path such that the bearing of the stationary emitter is held constant; the parameters of the flight path then give the emitter location without actually using the DF bearing. In this way, the moving receiver is free to concentrate on keeping an accurate fix on the emitter without needing to record exactly what this fix is. The down side to this is that a very special path geometry needs to be flown, which of course does not necessarily suit the goals of a given flight mission.

The effect of bearing bias in passive geolocation has also been studied by Gavish and Fogel [27], who consider a constant angle added to the emitter bearing. As we have done in Section 3.4, they define the notion of observability to mean that in the absence of noise but perhaps with this bias, the emitter's position is still able to be uniquely determined. They show that if there is any possibility of an additive bearing bias being present, then the receiver should not fly a circle that has any chance of passing though the emitter's position.

Their argument runs as follows. Suppose that to the right hand side of (3.41) we allow a constant bearing bias to be present, and disregard the noise so that we are only considering the systematic error of the added bias. Then the notion of observability means there is only one emitter position allowed as a solution to this equation. But suppose that on the contrary there are two distinct solutions, perhaps with different biases. Then it's straightforward to show that the track that gives these solutions is a circle passing through the emitter's actual position. So the moral is that if we want to be assured of not having any ambiguity, then we should not invite difficulties by flying any such circle in the first place.

Gavish and Fogel also consider the Cramér–Rao lower bound and how it is affected by a bias. In particular, they consider the case of a constant velocity receiver flying approximately broadside to an emitter. The bias added to the bearings is no longer a constant, but rather gaussian-distributed around zero mean with some specified standard deviation. They calculate a Cramér–Rao ellipse for a number of different track lengths, assuming a 2° standard deviation noise and both with and without a 3° s.d. bias. What they find is that, as might be expected, the error ellipse is larger for the biased case, and as a simple rule of thumb it could be said that typically the addition of the bias means that the major axis of the no-bias ellipse becomes the minor axis of the biased ellipse, with the shape of the ellipse being roughly unchanged.

Reports of Healy and of Beasley & Miles. Here we describe two longer reports that give an overview of stationary-emitter geolocation in general, as well as comparing different methods.

(1) Healy's thesis. The first is a thesis by Healy [28]. In this he compares the Gauss–Newton and RLS methods for the case of both stationary and moving emitters, with his main effort going toward quantifying the algorithms' effectiveness. After discussing the algorithms and the general problem to be solved, he runs each for a variety of scenarios and calculates several things: the speed of convergence, misadjustment accuracy (how well the algorithms pinpoint the emitter), computational complexity (number of floating point operations required for the calculations), and robustness to initial data and different conditions of data generation.

A selection of Healy's results have been verified or compared using the Matlab programme described in our Appendix C. One of Healy's initial results is that the speeds of convergence of G–N (Gauss–Newton) and RLS are similar once the RLS has overcome its slow start, during which time it needs to stabilise the matrix P in (3.45). In fact this is, however, quite dependent on how we implement the G–N algorithm. Healy divides the data points into batches of ten, overlapping by five, and runs the algorithm separately on each batch. In the Matlab programme of Appendix C, this has not been done; instead the algorithm is always run on the entire set of points since we have enough computational speed to justify this. What this means is that just one calculation is done at the end of the flight path. Doing this and recording the number of floating point operations required by Matlab to implement the various algorithms, it turns out that for fairly straightforward scenarios, Gauss–Newton takes about three times as many floating point operations as do both RLS and Kalman. Healy's results also agree with this.

On his page 77, Healy states that G–N's misadjustment accuracy is fairly well independent of its gain. By running our Matlab programme we find that although G–N does generally converge for each of the two gains that Healy uses (0.3 and 1), it might oscillate wildly and iterate a very large number of times before approaching a limit value. In agreement with Healy, we find that RLS is very sensitive to gain changes, as discussed in Appendix C.

Healy also finds that RLS and G–N are about equally robust when applied to various situations of moving emitters. On his page 87, Healy writes that the G–N algorithm's stability is not affected by the choice of gain. Although it is certainly true that changing the gain will not make the algorithm diverge, a change may well cause a convergence to a very different final value if the gain is too small, or cause it to oscillate heavily about the true value if the gain is too large—and also the numbers of iterations required for even these scenarios are heavily dependent on the gain choice. So it would be naïve to infer that just because an algorithm is stable, its results can be trusted.

Many of Healy's trials use a moving emitter, for which he has success with the RLS algorithm. Understandably his Gauss–Newton routine fails here, being only designed for the stationary case. In Section 3.8 we have tailored the G–N routine to cope with moving emitters.

(2) Report by Beasley and Miles. The second report available is by Beasley and Miles [2]. This provides a general overview of both direction finding and the associated statistical processing for geolocation, such as maximum likelihood. It then focuses on TDOA, using least-squares analysis with gaussian errors.

The report's modelling section describes the calculation of error ellipses, that are associated with geolocation from a constant velocity aircraft employing a small TDOA receiver at each wingtip. As expected, these ellipses show that errors are minimised for broadside emitters (the closer the better), with the down-range error increasing rapidly as the emitter angle moves away from broadside. This is a reasonable result, since the time differences of signal reception are then becoming less sensitive to distance. In that case too, the corresponding cross range errors also increase, but are much smaller in size. Perhaps the main results from the modelling section in [2] are that the down- and cross-range errors can be approximated as follows:

$$\text{Down range} \propto \frac{R^2}{Ld\sqrt{NB^3TP}}\,; \quad \text{Cross range} \propto \frac{R}{d\sqrt{NB^3TP}}\,, \tag{4.1}$$

where

$$\begin{aligned} R &= \text{range to emitter} & L &= \text{receiver flight path length} \\ d &= \text{baseline of sensor pair} & N &= \text{number of bearings taken} \\ B &= \text{emitter bandwidth} & T &= \text{length of signal wave train} \\ P &= \text{emitter's effective radiated power.} \end{aligned} \tag{4.2}$$

(Although these expressions are specifically for a TDOA scenario with two receivers, they are referred to on our page 46 in connection with a more general proposed rule of thumb.)

Also included in Beasley and Miles' report are studies of ways to beat systematic errors. One approach for estimating any bias uses reference emitters at known locations. Beasley and Miles do this, but for their example the method offers only a small improvement in accuracy. Another suggested way of cancelling bias is by the use of *race track* paths for the receiver, in which it flies along a straight line and then back again, with the aim of cancelling any errors introduced by the emitter's being always on the same side of the plane.

The report also discusses the physical characteristics required of the receiver antenna for good reception of the signal, and briefly discusses the relevant signal processing requirements. It finishes by describing DF work carried out by the United Kingdom Defence Evaluation and Research Agency (DERA) in Malvern, Defford, Portsdown, and Reading.

One area not well described in the geolocation literature is the shape of tracks flown, not just by emitters but also by receivers. Spingarn [3] and Challa & Faruqi [20, 29] use a constant velocity receiver with stationary emitter; Arulampalam [19] and Lindgren & Gong [30] use a zig-zagging receiver flying at constant velocity on each leg, following a constant velocity emitter. Farina [22] and Galkowski & Islam [31] do much the same but with a single change in receiver direction. This simplicity, especially in older literature, is presumably the result of the authors' not having fast enough computers at their disposal to be programmed easily for more complex motion. It is addressed by the Matlab programme described in Appendix C, which incorporates receivers and emitters flying arbitrary tracks—at least insofar as these are describable using a series of waypoints.

Moving Emitters. Much less space in the literature has been devoted to the more interesting subject of moving emitters, although several authors have discussed geolocating a constant velocity emitter. Lindgren and Gong [30] describe a Kalman filter with a simple receiver motion consisting of just two constant velocity legs, and find good convergence; although they concede that precisely how the receiver should move to maximise the algorithm's performance is a matter for further study. Lévine and Marino [32] discuss a similar problem, proving theorems about observability and the advantages of knowing the emitter's distance, but they carry out no simulations using actual data.

Other Kalman Filters. Arulampalam has published a report [19] in which he compares the performance of various Kalman filters. Besides the Cartesian and Modified Polar EKFs, he also considers others: the Pseudo-Linear Estimator EKF, Modified Gain EKF, and range-parametrised versions of the Cartesian and Modified Polar EKFs.

The first two of these other filters are built on modifications to x_k or H_k in Section 3.10. The range-parametrised versions of the CEKF/MPEKF are built essentially by running the usual CEKF/MPEKF filters in parallel for different range initialisations, and then combining the results in a way that uses bayesian analysis to provide weightings. This can all be computationally intensive, but in practice some of the parallel calculations have a negligible enough weighting as to exclude them from needing to be performed at all.

Arulampalam's work shows the following. The Pseudo-Linear Estimator is generally only as good as, or worse than, the others; although along with the MPEKF it outperforms the others in giving a heading error that quickly drops to zero with time. On the other hand it can give a poor range error. It is not considered further in this report. The performance of the Modified Gain EKF is generally similar to that of the CEKF and also is not considered further here. The two range-parametrised trackers are found to be of most efficacy only when there is little or no knowledge of the initial range. However, in Arulampalam's scenarios they are perhaps 50% better at estimating emitter speed than their CEKF/MPEKF counterparts, while their range, azimuth, and heading errors differ little from those of the usual CEKF/MPEKF. These range-parametrised algorithms have also been excluded from further study in this report.

Receiver Position Errors. Little work has been published on the case where the receiver positions are themselves subject to error. Ancker [8], in his rewrite of the Stansfield algorithm, considered the case where the linear errors in the receiver positions are gaussian and small compared to the emitter distances, but large enough to be comparable with linear errors corresponding to the bearing error. He calculated an effective bearing error by simply adding the two linear errors from the receiver position and the emitter location. This requires knowledge of the emitter range (as discussed previously), which in practice can be estimated initially and then again at each step of an iterative solution. Wax [33] considers the same problem and generalises it, although the analysis is somewhat obscure.

It is more realistic for the receiver positions not to have normally distributed errors, but rather to be subject to some offset. Receiver positions that are calculated using GPS have a very small error. More important is the gyro error of the inertial navigation system, since this cannot be calibrated using GPS. It can be modelled in a first analysis as increasing linearly, although it is also subject to an oscillation. But this error is also very small: a typical value for our two-dimensional scenario would be a 0.001° hr^{-1} increase in yaw, so that this would manifest as a bias of the same amount to each bearing measured. We have not considered such small values any further.

5 Geolocation For a Triangle of Receivers

In this section we consider a more restricted problem than previously. Given a geometry consisting of three stationary receivers placed in an equilateral triangle, using bearings-only geolocation for a stationary emitter, how good a geolocation can be done?

Our start point is a calculation of the Cramér–Rao bound, being the theoretical best-obtainable accuracy. The calculation is done here in detail for its pedagogical value. Next we run several simulations of various geolocation algorithms, and measure parameters such as the CEP. These parameters are then plotted against the emitter's distance from the triangle along the direction in which the Cramér–Rao results indicate that we can expect the worst estimates. Figure 5 shows the basic geometry always used.

5.1 Calculating a Cramér–Rao Set of Bounds

A set of error ellipses that describe the Cramér–Rao lower bound for a multiple receiver setup is produced in this section. As discussed in Section 3.2, the underlying principle is

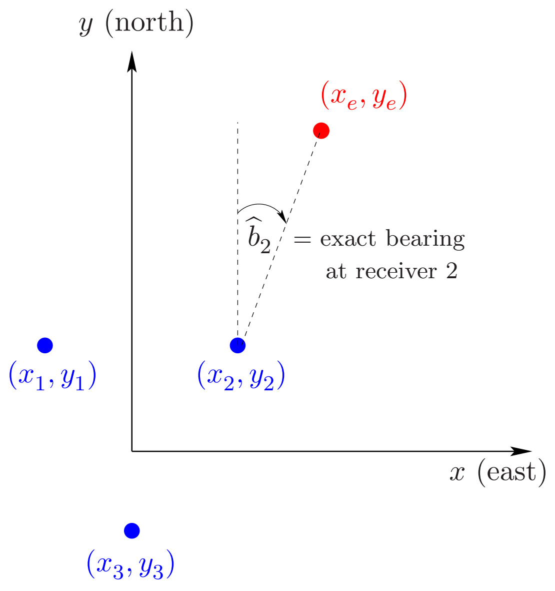

Figure 5: *Calculation of the Cramér–Rao bound for three receivers observing one emitter*

that the inverse of the Fisher Information matrix is the best covariance matrix obtainable for an unbiased estimator of the emitter location.

The Fisher Information matrix is produced in the following way. Suppose we have a set of n stationary receivers, each observing a stationary emitter (as in Figure 5, where $n = 3$). The exact bearing at receiver i is denoted $\widehat{b}_i$:

$$\widehat{b}_i = \tan^{-1} \frac{x_e - x_i}{y_e - y_i} + \text{quadrant-dependent constant.} \tag{5.1}$$

We define the actual (i.e. noisy) bearing observed by receiver i as b_i with error $\varepsilon_i \equiv b_i - \widehat{b}_i$. These errors are gaussian-distributed around zero with standard deviation σ_i:

$$p(\varepsilon_i) = \frac{1}{\sigma_i \sqrt{2\pi}} \exp \frac{-\varepsilon_i^2}{2\sigma_i^2} \,. \tag{5.2}$$

The procedure for calculating the Fisher Information matrix begins with the probability density for all of the receivers. They are independent, so we can write

$$p(\varepsilon_1, \ldots, \varepsilon_n) = p(\varepsilon_1) \ldots p(\varepsilon_n) \,. \tag{5.3}$$

As in (3.3), calculate the negative log-likelihood function defined by

$$L(x_e, y_e) = -\ln p(\varepsilon_1, \ldots, \varepsilon_n) = \sum_i \ln \left(\sigma_i \sqrt{2\pi} \right) + \frac{\varepsilon_i^2}{2\sigma_i^2} \,. \tag{5.4}$$

The Fisher Information matrix itself is then the expected value of a matrix of partial derivatives of this negative log-likelihood:

$$I \equiv \mathbb{E}\begin{bmatrix} \dfrac{\partial^2 L}{\partial x_e^2} & \dfrac{\partial^2 L}{\partial x_e \partial y_e} \\ \dfrac{\partial^2 L}{\partial x_e \partial y_e} & \dfrac{\partial^2 L}{\partial y_e^2} \end{bmatrix}$$

$$= \sum_i \frac{1}{\sigma_i^2} \begin{bmatrix} \mathbb{E}\left\{ \left(\dfrac{\partial \widehat{b}_i}{\partial x_e}\right)^2 - \varepsilon_i \dfrac{\partial^2 \widehat{b}_i}{\partial x_e^2} \right\} & \mathbb{E}\left\{ \dfrac{\partial \widehat{b}_i}{\partial x_e}\dfrac{\partial \widehat{b}_i}{\partial y_e} - \varepsilon_i \dfrac{\partial^2 \widehat{b}_i}{\partial x_e \partial y_e} \right\} \\ \mathbb{E}\left\{ \dfrac{\partial \widehat{b}_i}{\partial x_e}\dfrac{\partial \widehat{b}_i}{\partial y_e} - \varepsilon_i \dfrac{\partial^2 \widehat{b}_i}{\partial x_e \partial y_e} \right\} & \mathbb{E}\left\{ \left(\dfrac{\partial \widehat{b}_i}{\partial y_e}\right)^2 - \varepsilon_i \dfrac{\partial^2 \widehat{b}_i}{\partial y_e^2} \right\} \end{bmatrix}. \tag{5.5}$$

The expectations are taken with respect to the gaussian in (5.2)—which is of course an even function, in which case the following holds:

$$I = \sum_i \frac{1}{\sigma_i^2} \begin{bmatrix} \left(\dfrac{\partial \widehat{b}_i}{\partial x_e}\right)^2 & \dfrac{\partial \widehat{b}_i}{\partial x_e}\dfrac{\partial \widehat{b}_i}{\partial y_e} \\ \dfrac{\partial \widehat{b}_i}{\partial x_e}\dfrac{\partial \widehat{b}_i}{\partial y_e} & \left(\dfrac{\partial \widehat{b}_i}{\partial y_e}\right)^2 \end{bmatrix}. \tag{5.6}$$

The partial derivatives are found from (5.1), producing

$$I = \sum_i \frac{\begin{bmatrix} (y_e - y_i)^2 & -(x_e - x_i)(y_e - y_i) \\ -(x_e - x_i)(y_e - y_i) & (x_e - x_i)^2 \end{bmatrix}}{\sigma_i^2 \left[(x_e - x_i)^2 + (y_e - y_i)^2\right]^2}. \tag{5.7}$$

The Cramér–Rao theory states that I^{-1} is the best covariance matrix obtainable from an unbiased estimator of (x_e, y_e). The c-sigma uncertainty ellipse around each emitter position is then given by the following plot:

$$\begin{bmatrix} x - x_e & y - y_e \end{bmatrix} I \begin{bmatrix} x - x_e \\ y - y_e \end{bmatrix} = c^2. \tag{5.8}$$

If we take the scenario in Figure 5 with 1° bearings errors (i.e. all $\sigma_i = 1°$), then a plot of 3-sigma ellipses is shown in Figure 6. We see the expected threefold symmetry, as well as the fact that the best results (smallest ellipses) occur when the overall angular spread of the receivers is largest, when seen from the emitter (which lies at the centre of any given ellipse). Thus any emitter at the centre of the northernmost ellipses (and their counterparts at bearings of 120° and 240°) see the largest receiver-subtended angle, corresponding to these ellipses having the smallest major axes. The reverse is true for the southernmost ellipses and their counterparts. Minor axes comparisons are not so easy to make.

What we now wish to do is plot a representative dimension of, say, each 1σ Cramér–Rao ellipse for the worst case scenario (where two receivers are equally further distant than

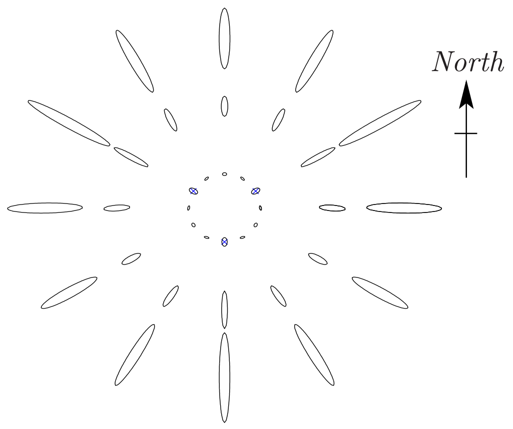

Figure 6: *Cramér–Rao 3σ ellipses for the setup of Figure 5. The receivers are blue crosses, with the centre of each ellipse being the actual position of the emitter used in the calculation of that Cramér–Rao ellipse. We have used 3σ instead of the more usual 1σ purely to make the ellipses large enough for easy viewing*

the closest receiver to the emitter), as a function of the emitter distance from the centre of the triangle. This worst case scenario corresponds to an emitter being at a bearing of 60° in Figure 6. What we will plot is the circular error probable (CEP), being the radius of the circle centred on the true emitter position, within which the emitter would be estimated to lie 50% of the time. This has been found by calculating the radius of the circle, centred at the emitter, that makes a domain such that when the double gaussian of the Cramér–Rao ellipse is integrated over this domain, the answer is 0.5. The resulting plots are in Figure 7 for bearing errors of 1° and 8°, along with baselines of 20 and 40 km.

5.2 Long-Baseline TDOA

In this section, we wish to establish an accuracy for geolocation using the long-baseline time difference of arrival technique, again for the above equilateral triangle of three receivers drawn in Figure 5. Each of these receivers is located at some distance R from the origin, and each has an inherent timing error of τ—meaning that the times they register are in error by an amount drawn from a normal distribution with zero mean and standard deviation τ.

Suppose that the signal arrives at receiver 1 at time t, at receiver 2 at time $t + \Delta t_{12}$, and at receiver 3 at time $t + \Delta t_{13}$. Then the radio waves have travelled at speed c for an extra distance $c\Delta t_{12}$ to get to receiver 2 as compared with receiver 1, and similarly for receiver 3. So if the unknown emitter location is (x, y), with the receivers at positions $(x_1, y_1) \to (x_3, y_3)$, then we can write, for $d_1 \to d_3$ being the distances of the receivers from the emitter:

$$\begin{aligned} d_i &= \sqrt{(x_i - x)^2 + (y_i - y)^2} \quad \text{for } i = 1 \to 3 \\ d_2 &= d_1 + c\Delta t_{12} \\ d_3 &= d_1 + c\Delta t_{13} \,. \end{aligned} \tag{5.9}$$

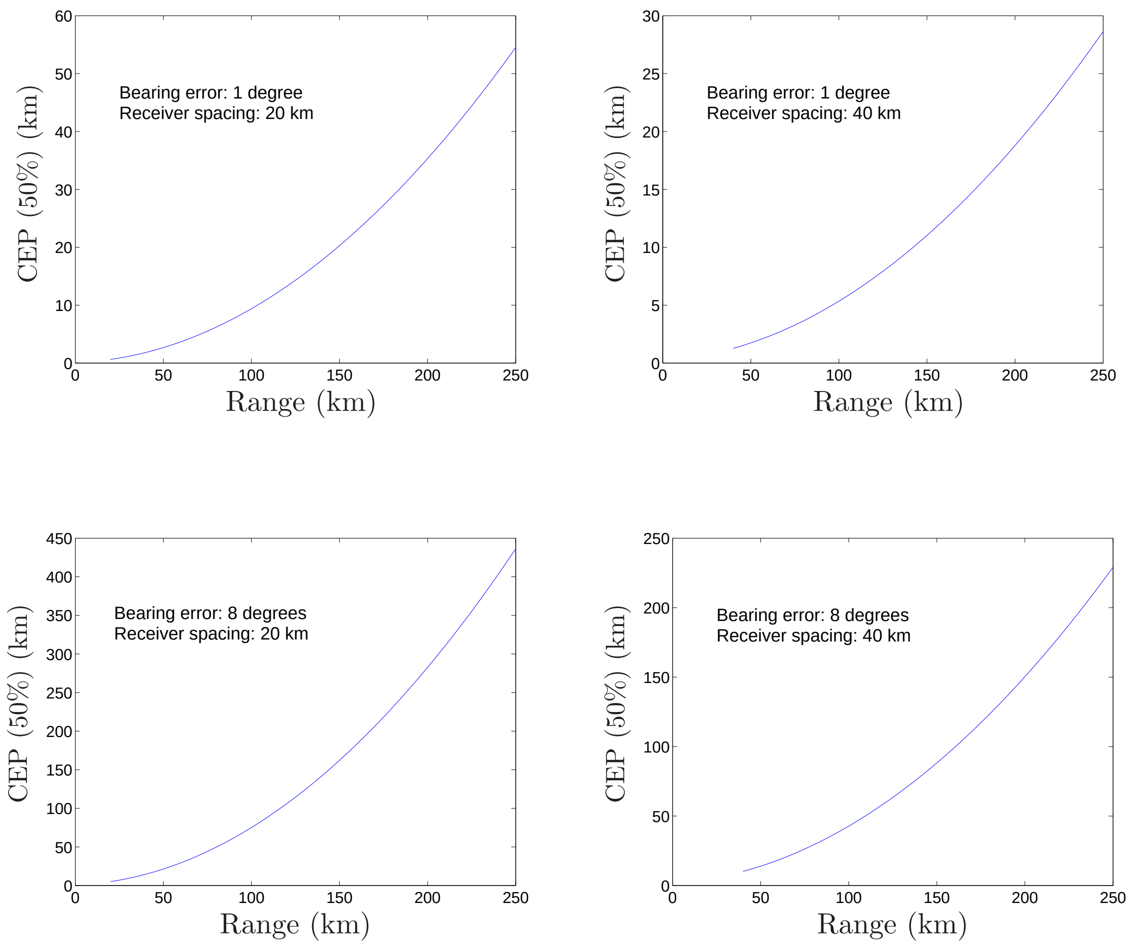

Figure 7: *50% CEP for Cramér–Rao estimates, applied to the setup of Figure 5, with the emitter at a 60° bearing as seen from the centre of the receiver triangle. Top row: bearing error 1°; baselines 20 km (left) and 40 km (right). Bottom row: bearing error 8°; baselines 20 km (left) and 40 km (right)*

Given time differences $\Delta t_{12}, \Delta t_{13}$, equation (5.9) can be solved numerically to find the emitter position. See the box on the facing page for how to do this.

Conversely, if we start with knowledge of the emitter position, then this TDOA can be used to establish a geolocation accuracy for this technique when the timing error τ is present: by simulating the geolocation many times and building up a spread of estimated positions for the emitter. First we place the emitter at some well defined position, and calculate values for $\Delta t_{12}, \Delta t_{13}$. Thus each geolocation simulation can be made by first perturbing these values, by adding to each of them a random number chosen from the normal distribution $N(0, \tau^2)$; and then treating the new "noisy" time differences as simulated data from which we calculate an estimated position for the emitter by solving (5.9). Doing this many times gives a spread of estimated emitter positions, and we can calculate a CEP for this spread, relative to the true position of the emitter.

Solving Nonlinear Simultaneous Equations

How do we solve the set of equations (5.9)? Write it as

$$\begin{aligned} d_2(x,y) - d_1(x,y) - c\Delta t_{12} &= 0 \\ d_3(x,y) - d_1(x,y) - c\Delta t_{13} &= 0 \end{aligned} \tag{5.10}$$

There are two equations in x and y here. With $\boldsymbol{x} \equiv [x\ y]^t$, write the first equation as $f_1(\boldsymbol{x}) = 0$ and the second as $f_2(\boldsymbol{x}) = 0$. For some $\boldsymbol{x_0}$ not too far from $\boldsymbol{x}$, we can Taylor-expand them both to first order, writing (and remembering that ∇f_1 and ∇f_2 are row vectors):

$$\begin{aligned} \begin{bmatrix} f_1(\boldsymbol{x}) \\ f_2(\boldsymbol{x}) \end{bmatrix} &\simeq \begin{bmatrix} f_1(\boldsymbol{x_0}) \\ f_2(\boldsymbol{x_0}) \end{bmatrix} + \begin{bmatrix} \nabla f_1(\boldsymbol{x_0})\ (\boldsymbol{x} - \boldsymbol{x_0}) \\ \nabla f_2(\boldsymbol{x_0})\ (\boldsymbol{x} - \boldsymbol{x_0}) \end{bmatrix} \\ &= \begin{bmatrix} f_1(\boldsymbol{x_0}) \\ f_2(\boldsymbol{x_0}) \end{bmatrix} + \underbrace{\begin{bmatrix} \nabla f_1(\boldsymbol{x_0}) \\ \nabla f_2(\boldsymbol{x_0}) \end{bmatrix}}_{\equiv J(\boldsymbol{x}_0)} (\boldsymbol{x} - \boldsymbol{x_0}) \end{aligned} \tag{5.11}$$

If $f_1(\boldsymbol{x}) = f_2(\boldsymbol{x}) = 0$ by definition, then

$$\begin{bmatrix} f_1(\boldsymbol{x_0}) \\ f_2(\boldsymbol{x_0}) \end{bmatrix} + J(\boldsymbol{x}_0)\ (\boldsymbol{x} - \boldsymbol{x_0}) \simeq 0\,, \tag{5.12}$$

or

$$\boldsymbol{x} \simeq \boldsymbol{x}_0 - J^{-1}(\boldsymbol{x}_0)\ \begin{bmatrix} f_1(\boldsymbol{x_0}) \\ f_2(\boldsymbol{x_0}) \end{bmatrix}. \tag{5.13}$$

This equation can be used iteratively: the initial estimate of the emitter position is $\boldsymbol{x}_0$, and the new estimate is given by the right hand side of (5.13).

When this is done using (5.13), we can get a feel for how the geolocation is a function of the baseline and timing accuracies. The timing accuracy is more crucial, by which is meant that if a given long baseline TDOA system has a certain geolocation accuracy, and we decide to halve the baseline, then in order to keep the stated geolocation accuracy, we must do more than halve the number specifying the timing accuracy. This is seen in the setup of Figure 5 (page 31), with a triangle side of 10 km and an emitter 100 km to the east. The receivers have a 1 ns timing accuracy. Following the procedure outlined in the previous paragraph, the 50% CEP is calculated to be 2.3 km. Now, if we reduce the triangle side length by a factor of ten to 1 km, does the accuracy required to keep the same CEP now also reduce by a factor of 10 to become 100 ps? No: the required timing accuracy turns out to be about 1 ps. So in order to reduce the size of the TDOA array, we need a drastically improved timing capability. This is shown in Figure 8.

Note that for TDOA, the geolocation accuracy is quite geometry-dependent. For the above case of a 10 km triangle side with 1 ns timing, if the emitter is 100 km *north* of the triangle (see Figure 9), then the 2.3 km CEP reduces to just 0.1 km. In that case, scaling the triangle down by a factor of 10 places less of a demand on the timing accuracy to keep the 0.1 km CEP: instead of 1 ps, we need just 10 ps.

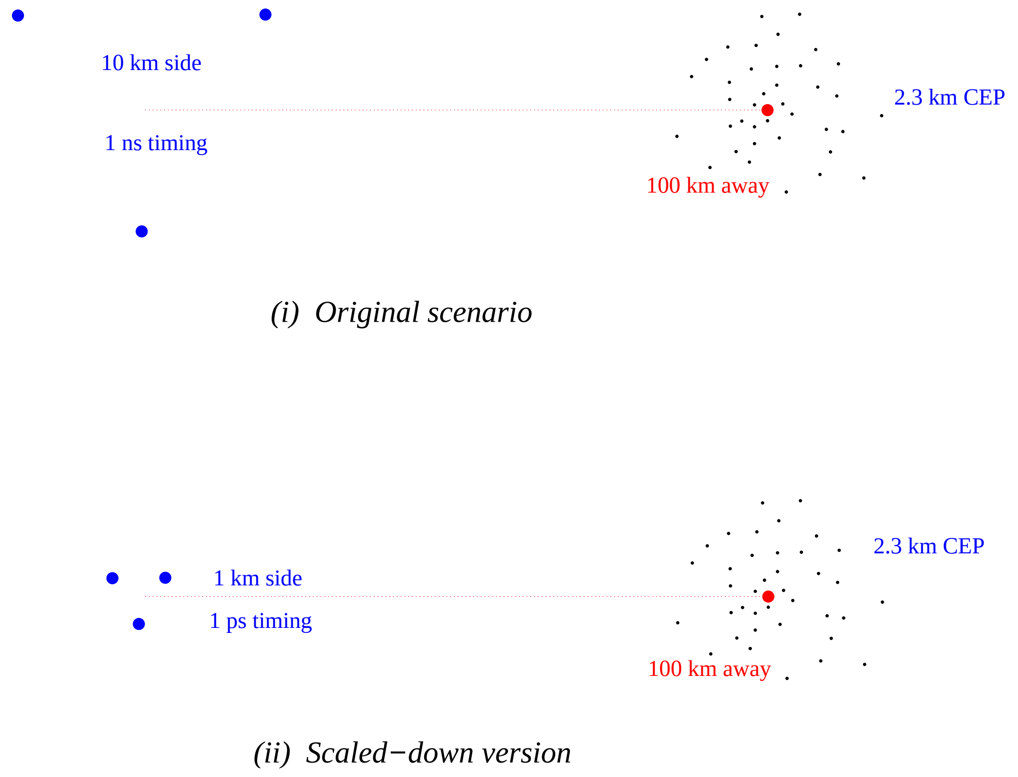

Figure 8: *A schematic of a Monte Carlo set of estimated emitter positions. Beginning with a 10 km triangle and 1 ns timing (i), the 50% CEP is 2.3 km. To maintain this CEP while scaling the receiver geometry down by a factor of 10 means increasing the timing accuracy by a factor of 1000, as seen in (ii)*

Linear Analysis

Although a linear analysis of TDOA might be possible, it is certainly fraught with difficulty. The reason is that the emitter is being located at the intersection of (at least) two hyperbolæ, and any linear assumption might affect the position of the distant arms of these that are doing the intersecting. Although it might be thought that as the emitter's distance increases, a linear assumption will be increasingly accurate, in practice a competing effect is present: the distance to the intersection of the appropriate hyperbolæ arms will become ever more sensitive to the position of those arms as they become more and more parallel. So even a slight error for a distant emitter can produce a huge error in the geolocation—even in the absence of noise.

A good example of this occurs in the following way. Suppose we first assume that the incoming rays from the emitter are parallel, and then use TDOA with simple trigonometry to establish a direction to the emitter. This method gives a very accurate emitter direction, but it cannot be used in practice to geolocate: although the second bearing line drawn with the aid of a third receiver does indeed intersect the first, the placement error (for no noise) turns out to be prohibitively large. For example, in the scenario of Figure 9, the emitter will be incorrectly geolocated at about twice its correct distance from the receiver

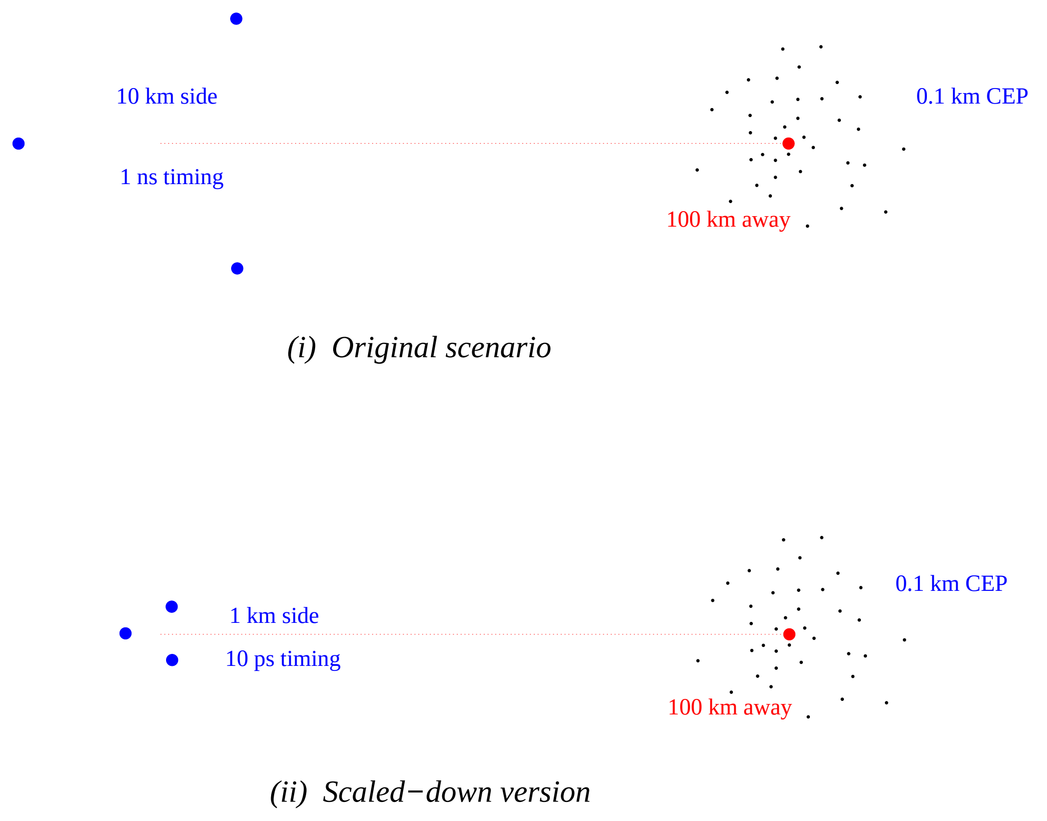

Figure 9: *As for Figure 8, but with a different receiver triangle orientation (i). The 50% CEP is now much smaller: 0.1 km. To maintain it now while scaling the receiver geometry down by a factor of 10 means we need "only" increase the timing accuracy by a factor of 100 (ii)*

triangle. In the final analysis, geolocation with TDOA certainly demands good baselines, timing, and numerical accuracy.

6 Results of Running a Matlab Simulation

This section describes some results of simulating various geolocation scenarios. These scenarios were used as a testbed for comparing the various geolocation algorithms described in this report. The algorithms were coded into Matlab using a graphical user interface, or "gui", designed to allow reasonably arbitrary scenarios to be set up. All scenarios assumed one emitter and one receiver, with either moving along a set of arbitrarily laid-down waypoints. The receiver would make a bearing measurement at time intervals that could also be specified and easily changed. These measurements were subject to bearing noise whose error could be input and changed easily. This gui is described in Appendix C.

The primary result to emerge from this process of running geolocation simulations is that rules of thumb for efficient geometries appear not to be obtainable. Simulations can always give a feel for what the algorithms are capable of, and what their foibles are; but simple rules of thumb have not arisen from any of the work described in this section. Because of that, the software is mainly useful for developing a feel for geolocation, and is available from the author. The package has survived a Matlab upgrade, although that upgrade created a bug that was only fixed with difficulty. Matlab software cannot be assumed to be backwards compatible. Also, any changes caused by an upgrade will not necessarily cause a graceful failure; rather, the software might still run, but give slightly wrong results—and then only sometimes.

The output from the following command is a simple example of how Matlab can give an empty-matrix result which is wrong due to (necessary) internal rounding, but which is meant to be "[3]": `find([0.5 : 0.1 : 0.8] == 0.7)`. It demonstrates that internal rounding errors do not necessarily imply a result that is incorrect in e.g. the tenth decimal place. Rather, they can be *completely* wrong, which the user of the software will not necessarily be aware of. Worse, very problematical to the code used in the gui is that there seems to be a new Matlab requirement to declare double precision in places not listed in its official documentation, and which was not the case in previous versions; and the explicit use of double precision is not commonplace in Matlab anyway. The problem is that not declaring double precision does *not* always simply lead to less accurate results; rather, it can produce meaningless ASCII characters in places where numbers are expected, which leads to erratic behaviour in the gui. (Submitting modified code for Matlab's `find` command, along with a request asking for the documentation to be updated, have not had any effect.)

Some care needs to be taken with coding geolocation routines as regards the trigonometry used. For example, Spingarn's well known work [3] uses an inverse tangent function; but such a function will need careful coding if it is not to fail in some situations. An example of the pitfall can be set up with two receiver trajectories that point in two only slightly differing directions, where for one the algorithm succeeds, while for the other it fails. It might be thought that since (3.41) uses an inverse tangent, then all we need do is call on the appropriate one of Matlab's two built-in functions for this purpose. The reason why this will sometimes fail is that the linearisation of the routines as in (3.35) expects the z-vector to have entries that become smaller with each iteration. But because these entries result from the difference of two angles, we might have a situation where one of the angles is a little more than zero (say 0.1 radians), while the other, given via an inverse tangent function, is a little under 2π: say 6.2 radians. While these two angles represent vectors that really are close to each other, their numerical difference is certainly nowhere near zero. So in practice, routines for dealing with angles often need more sophistication than Matlab's inverse tangent functions can provide. The gui software has been written to handle this situation.

As regards computing speed, recent increases in processor speed mean that the 3500 bearing computations that Healy mentions taking 15 minutes for RLS in 1996–97 [28] now take just a few seconds on a standard personal computer. This is a good indication that geolocation algorithms can now be more realistically used than they could even a few years ago.

In an effort to investigate the parameters used by Healy for the Gauss–Newton and RLS routines, the programme used in this report was first run using a fairly straightforward initial scenario: a 50° subtended receiver flight path as seen by the emitter, emitter broadside to mid-path, flight path length 100 units, initial estimate of emitter position about 20 units from actual position and somewhat closer to the flight path than the emitter. A very convoluted flight path with multiple loops and curves was also used. Bearing error was set at 1–2°, which is a very typical figure. What resulted were the following parameter choices:

Gauss–Newton gain Healy [28] determines the best value by asking how many iterations are required in order for the difference in successive iterations to be less than 0.001. He finds that a gain of 1 gives the fastest convergence, although apparently has not considered gains higher than 1. When running the Matlab programme we find that for a bearing noise of 1°, the number of iterations required to get to a < 0.001 update is minimised for gain values in the range 0.95–1.05, and hence when running the programme the gain has usually been set equal to 1.

RLS parameters There are two that need determining (see page 20):

1. The constant b. The value of this is not well discussed in the literature, but is supposed to be large in some sense. Healy suggests a value of between 1 and 10, but eventually uses both 150 and 0.1 with a moving emitter. The value of b needs to be set in conjunction with the following parameter λ.
2. The value of λ. Healy sets this equal to $1.65k/(k+1)$ where k is the current iteration number. If we do this and use his value of $b = 0.1$ with 50 bearings, we find that RLS only converges when the region of interest is small—so it appears that the values must be chosen to reflect the typical distances between the various points. So a range of b's and λ's was tested. When $b = 1$, the final estimate of emitter position oscillates wildly for say $\lambda \lesssim 0.4$, settles to within the 5σ uncertainty ellipse for $\lambda = 0.65 \to 0.70$, and then falls quite short as λ increases: for $\lambda = 0.72$ the routine barely arrives at the 30σ ellipse.

   As b is increased over the values $10, 10^2, 10^3$ up to 10^8, the typical λ values for which the iterations converge to within the 5σ ellipse or better move steadily upward through the values 0.7, 0.8 etc., but the estimated emitter positions still fall short as $\lambda \to 1$, unless $b \gtrsim 10^6$. By the time $b \gtrsim 10^7$, then for $\lambda \gtrsim 0.95$, the approximation to even 2σ or better is very good—and it remains very good when the region over which the scenario is played out shrinks to the size for which Healy's values $b = 0.1, \lambda = 1.65k/(k+1)$ were usable above. Hence the values chosen for the following simulations were $b = 10^8, \lambda = 1$.

Further preliminary notes to emerge from running the programme are summarised in the following paragraphs.

CPLE approach. This gives quite accurate initial estimates for the case of a stationary emitter. For example, if the flight path containing ten data points (each with 2° bearing noise) subtends an angle of 15° at the emitter, then typically the emitter position is calculated as fairly evenly spread over a 2σ Cramér–Rao ellipse, with a final down-range error of about 2%, and a final cross-range error of <1%.

The method also gives excellent estimates of initial position, velocity and acceleration for an emitter of constant acceleration, provided no more than about 0.1° bearing noise is present. Beyond this it loses accuracy quickly. The more we ask of our bearing data in terms of showing higher derivatives of position, the more sensitive the results are to noise.

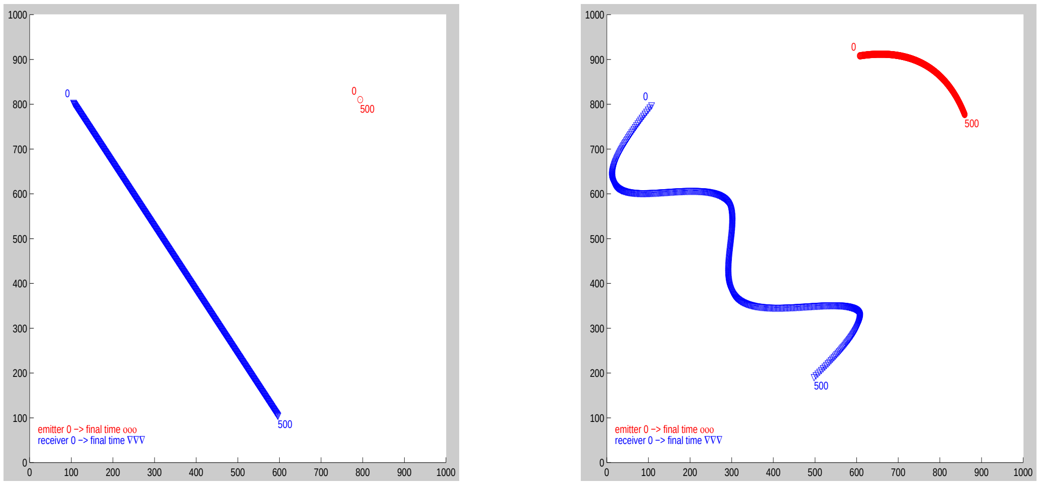

Figure 10: *Scenarios 1 (left) and 2 (right) referred to on page 40*

Kalman covariance matrix. Although the Cartesian Kalman covariance matrix $P_{k|k}$ represents the uncertainty in the state estimate, in practice its entries might be huge even when the estimates are quite close to the actual emitter position, or they might be small even though the routine is not locating the emitter very well at all. So they seem not to be useful, except for the fact that $P_{k|k}$ is used to calculate the next state estimate, and this can sometimes be very accurate even though $P_{k|k}$'s entries are anomalously large. Presumably the failure of $P_{k|k}$ to adequately reflect the true situation is related to the use of linearisation in the extended Kalman filter.

Cramér–Rao lower bound. This is usually, if not always, much better than the typical accuracies achieved by RLS and Kalman, although Gauss–Newton does tend to give accuracies around the Cramér–Rao mark.

Algorithm speed and number of computations. The various algorithms can be quantified more fully, to measure how long they take and how many computations they each perform. Two simple scenarios are shown in Figure 10, which shows how these are plotted within the Matlab gui.

Scenario 1: Use a stationary emitter, and a receiver flying at constant velocity and taking 500 bearings. Model the emitter as stationary.

Scenario 2: Use an emitter moving with constant speed but in a slightly curved path, with the receiver following a more zig-zag motion. This more complicated receiver motion must be chosen because the receiver needs to outmanoeuvre the emitter in order for the geolocation to work. Again take 500 bearings. We try various models:

- **a:** Emitter modelled as stationary.
- **b:** Emitter modelled as accelerating, with initial conditions supplied that are not very accurate.
- **c:** Emitter modelled as accelerating, with very accurate initial conditions supplied.

For these runs we have used the following parameters (refer to Figure 10 for the scale): initial position estimate = $(600, 700)$; initial velocity estimate = $(2, 0)$; initial acceleration estimate = $(0, 0)$; receiver x, y position errors taken to be zero for all the routines, and:

CPLE: Has no parameters.

G–N: Gain = 1, tolerance = 5 (i.e. this is the distance between subsequent emitter position estimates that, once achieved, causes the algorithm to stop).

RLS: $\lambda = 1, b = 10^8$ as explained on page 39.

CEKF: $P_{0|0} = 1000 \times$ unit matrix, $Q =$ unit matrix.

MPEKF: Initial emitter distance error $\sigma_r = 100$; initial emitter speed error $\sigma_v = 5$; initial range $\simeq 500$. This routine is found to be grossly unstable if we model the emitter's acceleration by adding any sort of Q, so we have set matrix Q equal to zero.

The performance of the various routines is presented in Tables 1–4 on the next page. Each main entry refers to a 1° bearing error, with the corresponding result for a 0.1° bearing error in parentheses. What do these results tell us?

CPLE handles stationary emitters well. It handles accelerating emitters well only if they are modelled correctly, but even then, is easily thrown out by bearing noise. It is fairly fast to run.

Gauss–Newton takes the longest time to process the data, but this is obviously affected by how strongly we require it to converge, by stopping its iterations when the final result changes by less than some tolerance. It locates stationary emitters well. Like the CPLE, its performance degrades if we model the emitter incorrectly. Although fairly insensitive to our guess of the emitter's initial position, it is badly affected by poor estimates of the initial velocity and acceleration.

It should be remembered that this G–N calculation is being done from one position of the receiver (the last), using all of the available data, and so is an estimate made at that position only. On the other hand, RLS and Kalman give one estimate for each flight path point while the receiver is moving. Even so, *their* estimates do not always settle to any obvious limit point at all. In fact these estimates might wander about significantly before apparently beginning to settle in the vicinity of the actual emitter position.

RLS performs well for a stationary emitter in the presence of noise. When the emitter moves, good results are only obtained if we model it correctly, and then RLS is not greatly affected by a poor choice of initial conditions. However it can sometimes diverge, even for the low bearing error case chosen of 0.1°.

Cartesian EKF only performs well if the emitter is modelled loosely as stationary, although the addition of a non-zero Q means that we are actually modelling a certain amount of unknown behaviour in its motion. In fact, it appears to be more effective to include a nonzero Q to model acceleration, than it is to explicitly model the emitter as accelerating by altering F in (3.46).

Table 1: *Performance comparison of the different routines: Scenario 1 (page 40). Main entry is for 1° bearing error; parentheses refer to result for 0.1°*

	CPLE	G–N	RLS	CEKF	MPEKF
Final cross range error (% of final range)	0.11 (0.01)	0.04 (0)	0.3 (0)	0.12 (0.03)	1.6 (31)
Emitter placement error (% of final range)	0.2 (0)	0 (0)	0 (0)	5 (10)	41 (430)
cpu time (s)	1 (1)	3.7 (3.7)	1.5 (1.5)	1.2 (1.2)	2.5 (2.5)
kflops	23 (23)	81 (81)	50 (50)	59 (59)	433 (433)

Table 2: *Performance comparison of the different routines: Scenario 2a (page 40). Main entry is for 1° bearing error; parentheses refer to result for 0.1°*

	CPLE	G–N	RLS	CEKF	MPEKF
Final cross range error (% of final range)	2.3 (2.1)	2.0 (1.9)	3 (3)	0.6 (0.1)	16 (160)
Emitter placement error (% of final range)	55 (56)	57 (57)	55 (56)	8 (3)	50 ± 50 (400)
cpu time	1 (1)	4.5 (4.6)	1.5 (1.5)	1.2 (1.2)	2.5 (2.5)
kflops	23 (23)	99 (99)	50 (50)	59 (59)	433 (433)

Table 3: *Performance comparison of the different routines: Scenario 2b (page 40). ("div." = "diverges".) Main entry is for 1° bearing error; parentheses refer to result for 0.1°*

	CPLE	G–N	RLS	CEKF	MPEKF
Final cross range error (% of final range)	4 (0.05)	diverges (diverges)	0.3 ± 0.2 (0.06)	div. (div.)	div. (div.)
Emitter placement error (% of final range)	55 (2)	div. (div.)	100 ± 30 (1.5)	div. (div.)	div. (div.)
cpu time	1.2 (1.2)	div. (div.)	2.2 (2.2)	1.6 (1.6)	2.5 (2.5)
kflops	106 (106)	div. (div.)	480 (480)	865 (865)	433 (433)

Table 4: *Performance comparison for Scenario 2c (page 40). Main entry is for 1° bearing error; parentheses refer to result for 0.1°*

	CPLE	G–N	RLS	CEKF	MPEKF
Final cross range error (% of final range)	4 (0.05)	0.02 (0.01)	3 ± 2 (0.02)	div. (div.)	0.7 (0.3)
Emitter placement error (% of final range)	55 (2)	3.4 (0)	100 ± 200 (0.3)	div. (div.)	17 (10)
cpu time	1.2 (1.2)	2.8 (1.6)	2.3 (2.2)	1.6 (1.6)	2.5 (2.5)
kflops	106 (106)	240 (140)	480 (480)	865 (865)	433 (433)

***Table 5:** Performance comparison of the different routines*

	CPLE	G–N	RLS	CEKF	MPEKF
Need initialising?	no	yes	yes	yes	yes
Correct convergence (low noise)?	v. good	good	v. good	poor	good
Robust? Stationary emitter:	yes	yes	yes	no	no
Robust? Moving emitters:	yes	no	moderate	no	no
Heavily affected by noise?	yes	no	no	no	no

Modified polar EKF This is very sensitive to a good choice of initial conditions, as was also found by [19]. Although in principle capable of good accuracy, this routine needs a lot of computation even for the simple Scenario 1 on page 40. But certainly the nature of the computations is also important: for example, note that in Scenario 2b the Cartesian EKF takes only 1.6 seconds, but uses many more kiloflops (kilo floating point operations) in Matlab than the other routines; those mostly take longer to run but use fewer kiloflops. The main improvement of Modified Polar over Cartesian EKF is that MPEKF is far less sensitive to how we model the emitter's motion. Good results are obtained even if we model a constant velocity emitter as having constant acceleration, whereas the CEKF would perform very badly in that situation.

In general, for Scenario 1 on page 40 (stationary emitter), CPLE and G–N converge better to the actual position than do either RLS or Kalman. CPLE does not need initialising, while G–N is extremely robust and will usually converge to the correct position when the initial estimate is so bad that RLS and Kalman give useless results. Extremely noisy bearings usually throw CPLE, RLS, and Cartesian Kalman completely off track, and though they might still converge to some limit point, it will almost certainly be nowhere near the actual emitter position. In the same case G–N generally will still converge very accurately, although it might take longer than usual to do this. Broadly speaking, for a one-off estimate of where the emitter is, G–N gives better results than RLS/Kalman, while if we are interested in fast real time tracking, then RLS/Kalman are better suited if they are provided with a good initial state estimate. This estimate can easily be provided by the CPLE approach. These results are summarised loosely in Table 5, although it is very difficult to draw general conclusions.

6.1 Reproducing Some Known Results

In this section, we run the Matlab programme to check on some known results and as a further comparison of the different routines. The results we are seeking to reproduce are samples taken from graphs in [34]. The emitter-receiver layout as used in this reference is shown in Figure 11. In the four different scenarios reproduced here, the receiver always moves at 105 m/s westwards (204 kts) with each flight lasting for 350 seconds, taking one bearing every 50 seconds. In all cases the receiver position is known with zero error.

Reference [34] does not specify the algorithm it uses, describing it only as a simple least-squares routine. It quantifies this routine's accuracy for each geometry by doing 500 runs—each run giving one estimate of the emitter position—which are then averaged, producing

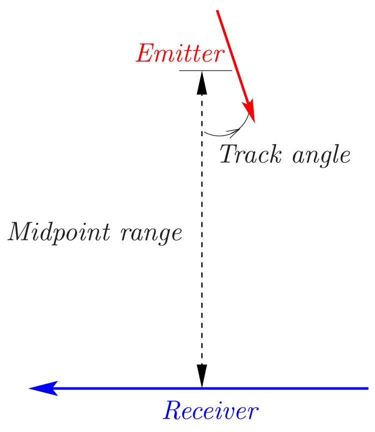

Figure 11: *Typical geometry used in Section 6.1*

the mean deviation of the estimated emitter position from its actual position (or its last position if it is moving). We have done likewise using the Matlab programme. Additionally, the Matlab programme produces a 50% CEP that gives a simple one-parameter indication of the spread of the estimates. In the Matlab programme each scenario uses only 100 runs, since this number already gives good statistics.

The four scenarios chosen were as follows:

1. Emitter stationary (at the middle of the path drawn in Figure 11); bearing error 1°. Range from midpoint of receiver path to emitter is 50 km.

2. Emitter moves at 10 m/s at a track angle of 30°; bearing error 1°. Midpoint range 50 km.

3. Emitter moves at 20 m/s at a track angle of −60°; bearing error 1°. Midpoint range 50 km.

4. Emitter moves at 20 m/s at a track angle of 45°; bearing error 2.3°. Midpoint range 200 km.

For each scenario we first run the CPLE routine, always modelling the emitter as stationary even when it is moving, as per [34]. As usual, we use CPLE to seed the other routines. The only exception to this is the modified polar Kalman filter, which is fairly sensitive to initial conditions, and the forms it requires are not quite given by the CPLE routine. In particular, MPEKF needs an initial speed error estimate which CPLE cannot provide, since we are necessarily modelling the emitter as stationary. This error has been specified somewhat arbitrarily by us.

The results for the mean deviation of the estimated emitter from the actual emitter (or its last position if it is moving) are given in Table 6. The report's values were taken from its graphs, while the values output from the Matlab programme include a 50% CEP in parentheses.

In general, the spread of estimates tends to overlap the actual emitter position. But this is by no means always the case, because of varying sensitivities to initial estimates; in particular the fact that we are modelling the emitter as stationary means that we are always

Table 6: *Comparison of performance values of [34] with the Matlab programme described in this report. Shown are results for the mean deviation of estimated emitter position from actual emitter position (or its last position if it is moving). Values in parentheses are a 50% CEP. All distances are in kilometres*

Scenario (page 44)	Report [34] (unknown algorithm)	Matlab programme modelling:				
		CPLE	G–N	RLS	CEKF	MPEKF
1	1.4	0.12 (1.1)	0.14 (1.0)	0.24 (0.9)	0.10 (0.8)	1.4 (4.6)
2	1.5	1.1 (1.4)	0.9 (1.3)	0.9 (1.2)	1.1 (1.1)	6.1 (5.5)
3	12	11 (11)	12 (12)	12 (12)	13 (13)	3 (6)
4	63	56 (61)	8 (33)	47 (36)	43 (42)	99 (66)

estimating its velocity to be zero, and this can badly degrade some routines' performance. Further simulations with the emitter's constant velocity being modelled cannot be done in this scenario, because the receiver never accelerates, and so it can never geolocate a moving emitter. Hence we have not added any results for the emitter modelled as moving, since that requires a remodelled receiver path, which then complicates or destroys the comparison. But the performances of the Matlab routines do generally agree with those of [34], which is what we set out to test.

The sensitivity of MPEKF to our choice of initial conditions is worth noting. The "99 (66)" values correspond to an initial range of 500 km, range error of 100 km, and speed error of 5 km/s (rather large, admittedly). If we change these values to 200, 50 and 0.1 respectively the tabulated values become 63 (64). If we now change the initial conditions to 200, 10, 0.5, the tabulated values become 1.3 (46). These tabulated values will also change from one set of 100 runs to the next. Clearly this sensitivity implies possible low expectations for our results if we start with no prior knowledge of the initial conditions.

6.2 Checking an Old Suggested Rule of Thumb

Reference [34] writes down rules of thumb for computing expected cross- and down-range errors for the case of a stationary emitter and a receiver moving with constant velocity. It includes no derivations, and its two rules appear to be based only on a vague but incorrect geometry. Certainly its expression for relative cross-range error is not reasonable, and does not work in practice.

If the cross- and down-range errors are labelled σ_{CR}, σ_R respectively, with k bearings over a flight path that sweeps out an angle θ_s as seen from the emitter (as in Figure 12), with a bearing error of σ_b, then the rules of thumb are given to within a factor of about unity as

$$\frac{\sigma_{CR}}{R} \simeq \frac{\sigma_b}{\sqrt{k}} \qquad\qquad \frac{\sigma_R}{R} \simeq \frac{2\sigma_b}{\theta_s \sqrt{k}} \,. \tag{6.1}$$

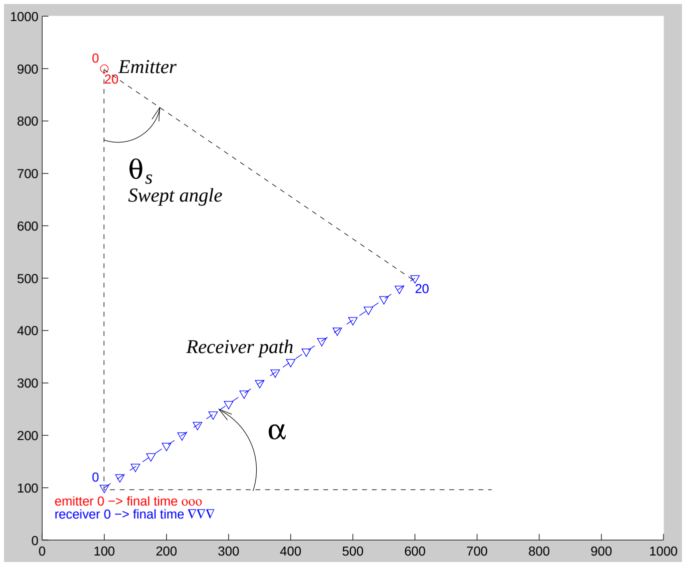

Figure 12: *Typical scenario tested in Section 6.2*

Note that the relative cross-range error is independent of swept path, something we will show to be false. This is not a simple mistake in [34], because the relative cross-range error is plotted there quite explicitly as a constant.

Expressions (6.1) are reminiscent of Beasley and Miles' result (4.1), which has

$$\frac{\sigma_{CR}}{R} \propto \frac{1}{\sqrt{k}} \qquad\qquad \frac{\sigma_R}{R} \propto \frac{1}{\theta_s \sqrt{k}} \,. \tag{6.2}$$

However Beasley and Miles' result is calculated only for a very specific case of TDOA with two receivers, and so cannot include any bearing error σ_b. Their cross-range error also comes out as independent of swept angle θ_s.

The scenario used in the present report to test the rules in (6.1) is shown in Figure 12. A stationary emitter is located at (100, 900), with a receiver flying at constant velocity, starting at (100, 100). The end point of the receiver flight path is placed variously over the grid in such a way that the swept angle θ_s and the angle α are varied over a convenient range. For each of the end points these two numbers produce, we consider a range of bearing errors and a range of numbers of bearings taken.

Each single scenario thus resulting is then repeated 100 times, with new noise randomly added in each run for a Monte Carlo approach. This yields average measured values of σ_{CR}/R and σ_R/R, together with the corresponding predictions obtained from (6.1).

As well as the relative cross- and down-range errors, the circular error probable (CEP) is also important, since such a figure is necessary for deciding whether a missile can pass close enough to the emitter for its explosive radius to be sufficient to have an effect.

Comparison with Rules of Thumb ("r.o.t."):

CPLE As the number of bearings taken increases, σ_R/R increases from $\sim 0.2 \times$ rule of thumb to $\sim 2 \times$ r.o.t., while σ_{CR}/R increases from ~ 0.25 r.o.t. to 5 r.o.t.

G–N σ_R/R drops from 0.15 r.o.t. to a very small fraction if α is small, but stays roughly constant at 0.15 r.o.t. if α is not small; σ_{CR}/R is always about 0.20 r.o.t., dropping slightly as the number of bearings increases.

RLS σ_R/R drops from 0.15 r.o.t. to < 0.05 r.o.t; σ_{CR}/R is always at 0.30 r.o.t., perhaps dropping slightly with an increasing number of bearings.

CEKF σ_R/R increases from 0.20 r.o.t. to r.o.t; σ_{CR}/R is around 0.40 r.o.t. but can fluctuate wildly.

MPEKF σ_R/R drops from r.o.t. to perhaps 0.1 r.o.t; σ_{CR}/R drops from 0.50 r.o.t. to perhaps 0.05 r.o.t. if $\alpha = 0$, but can grow to 2 r.o.t. if $\alpha = 60°$.

Clearly the trends are not especially simple or linear.

Least squares fitting of the data. Consider, say, σ_{CR}/R as a function of σ_b, θ_s, α, and the number of bearings n. We wish to try to fit some sort of function of the four parameters to the data, in a least-squares sense. A cue for what function to use is taken from the fact that the data is supposedly at least roughly fitted by (6.1). So perhaps a polynomial in $\sigma_b, \theta_s, \alpha$, and $\sqrt{n}$ is needed. This immediately presents a problem, because even a simple case can have many terms. A "simple" fitting function might be:

$$\sigma_R/R = z_1\sigma_b + z_2\theta_s + z_3\alpha + z_4\sqrt{n} + z_5\sigma_b/\theta_s + \ldots + z_{21}/\sqrt{n}\,, \tag{6.3}$$

where the highest order of any parameter is just one, terms with one parameter in numerator and denominator are included, and terms of order -1 are included since we are fitting a surface that looks to have something like a hyperbolic shape. The calculation can be done easily using Matlab, but a good fit does not result: although many data points might fit within a 10% margin, others will be out by a factor of 2. Of course higher order fitting surfaces can be tried, but the number of terms required to describe these becomes very large. Even using (6.1) as a cue to fitting with just one parameter,

$$\frac{\sigma_R}{R} = z_1 + z_2\frac{\sigma_b}{\theta_s\sqrt{n}}\,, \tag{6.4}$$

does not give a good fit over the whole surface, and experimenting with higher order polynomials in $\sigma_b/(\theta_s\sqrt{n})$ does no better. So it's not apparent that any useful rule of thumb can be arrived at, even with such a simple geometry as tried here: a constant velocity receiver and a stationary emitter.

7 Concluding Remarks

The search for straightforward rules of thumb for expected geolocation accuracies is a difficult one, at least in the sense of writing down an expression that takes in the parameters of a platform's motion, and produces the various geolocation errors for the various techniques. The search depends on which technique is used, and also depends heavily on receiver geometry and the kinematics of emitter/receiver motion.

In the preceding pages, we have described tackling the problem by first explaining common geolocation techniques. Various details of theory and implementation have been given, so that the algorithms can be run with little extra research required of the reader.

Although we have calculated some Cramér–Rao bounds, being the theoretical best possible accuracy obtainable for a given setup, we have tended to take a sort of "hands-on" statistical view of quantifying the accuracies. That is, for this report Matlab software was developed that simulates various scenarios, and this was used in a Monte Carlo approach to build up many estimates of where an emitter is, based on gaussian noise added as bearing error. Initial work was hampered by relatively long computing times, since much was being asked of the software. Currently, even on a time scale of one or two years, computer speeds increase enough that more and more processing can readily be done in geolocation algorithms.

In the final analysis, perhaps it might be that the best way to predict the accuracy that a geolocation algorithm can achieve, is to model a required scenario fairly specifically. In this respect, attempts to optimise a receiver configuration from the outset in any purely theoretical way appear to be very difficult—especially without depending on some sort of visualisation tool such as described in this report.

References

1. D.J. Torrieri (1984), Statistical theory of passive location systems, *IEEE Trans. Aero. Elect. Syst.*, 183–198
2. P.D.L. Beasley, D.A. Miles (1998), Enhanced geolocation for electronic reconnaissance, DERA
3. K. Spingarn (1987), Passive position location estimation using the extended Kalman filter, *IEEE Trans. Aero. Elect. Syst.*, 558–567
4. K.L. Brown (1997), Emitter position estimation from N independent bearing measurements, unpublished, DSTO
5. M.S. Arulampalam, N. Gordon, S. Maskell, W.J. Fitzgerald (April 2002), Bistatic radar tracking using bearing and Doppler measurements, DSTO report DSTO-TR-1303
6. P. Tichavský, C.H. Muravchik, A. Nehorai (May 1998), Posterior Cramér–Rao bounds for discrete-time nonlinear filtering, *IEEE Trans. Sig. Proc.* **46**, 1386–1396
7. R.G. Stansfield (1947), Statistical theory of DF fixing, *IEE Proc. Radar, Sonar & Nav.*, 762–770. See [8] for a clearer exposition.
8. C.J. Ancker (1958), Airborne direction finding—the theory of navigation errors, *IRE Trans. Aero. Nav. Elect.*, 199–210
9. *Kalman Filtering: Theory and Application*, ed. H.W. Sorenson. IEEE Press (1985)
10. W.H. Press et al. (1995), *Numerical Recipes in C*, 2nd edition. Cam. Uni. Press

11. W.H. Foy (1976), Position-location solutions by Taylor series approximation, *IEEE Trans. Aero. Elect. Syst.*, **12**, 187–194

12. R.L. Plackett (1950), Some theorems in least squares, Biometrika **37**, 149

13. Shaohua Niu, D.G. Fisher, Deyun Xiao (1992), An augmented UD identification algorithm, *Int. J. Control*, **56**, 193–211

14. R.E. Kalman (1960), A new approach to linear filtering and prediction problems, *J. Basic Eng.* **82D**, 35–45

15. P. Strobach (1990), *Linear prediction theory: a mathematical basis for adaptive systems*, Springer

16. R.J. Elliott, V. Krishnamurthy (1999), New finite-dimensional filters for parameter estimation of discrete-time linear gaussian models, *IEEE Trans. Automatic Control* **44**, 938–951. See also Exact finite-dimensional filters for maximum likelihood parameter estimation of continuous-time linear gaussian systems, *SIAM J. Control Optim.* **35**, 1908–1923 (1997); W.P. Malcolm, R.J. Elliott, A general smoothing equation for Poisson observations, obtained from R.J. Elliott

17. D. Koks, S. Challa (2003), An Introduction to Bayesian and Dempster–Shafer Data Fusion, DSTO report no. DSTO-TR-1436. Also S. Challa and D. Koks (April 2004) Bayesian and Dempster–Shafer fusion, *Sādhanā* **29**, 145–174.

18. R.F. Berg (1983), Estimation and prediction for maneuvering target trajectories, *IEEE Trans. Automatic Control* **28**, 294

19. S. Arulampalam (2000), A comparison of recursive style angle-only target motion analysis algorithms, DSTO report no. DSTO-TR-0917. Note there is a sign error in his equation (38): the last line should be

$$\alpha_4 = y_2(k) - y_4(k)[u_3(k, k+1)\sin\beta(k) + u_4(k, k+1)\cos\beta(k)]$$

20. S. Challa, F.A. Faruqi (1996), Non-linear system/linear measurements approach to passive position location using extended Kalman filtering, *IEEE TENCON Digital signal processing applications*, 665–669

21. V.J. Aidala, S.E. Hammel (1983), Utilization of modified polar coordinates for bearings-only tracking, *IEEE Trans. on Automatic Control* **28**, 283–294

22. A. Farina (1999), Target tracking with bearings-only measurements, *Signal Processing* **78**, 61–78. See also A. Farina (1998), Association of active and passive tracks for airborne sensors, *Signal Processing* **69**, 209–217

23. J.L. Poirot, G.V. M^{c}Williams (1974), Application of linear statistical models to radar location techniques *IEEE Trans. Aero. Elect. Syst.*, **10**, 830–834

24. K.D. Rao, D.C. Reddy (1994), A new method for finding electromagnetic emitter location, *IEEE Trans. Aero. Elect. Syst.*, **30**, 1081–1085

25. M. Gavish, A.J. Weiss (1992), Performance analysis of bearing-only target location algorithms, *IEEE Trans. Aero. Elect. Syst.*, **28**, 817–828

26. P.R. Mahapatra (1980), Emitter location independent of systematic errors in direction finders, *IEEE Trans. Aero. Elect. Syst.*, **16**, 851

27. Motti Gavish, Eli Fogel (1990), Effect of bias on bearing-only target location, *IEEE Trans. Aero. Elect. Syst.*, 22–25

28. G.J. Healy (1996–1997), Tracking algorithms for passive detection applications, MSc in School of Engineering and Applied Science, Royal Military College of Science, Shrivenham

29. S. Challa, F.A. Faruqi (1997), Passive position location using Bayes' conditional density filter, *SPIE* **3087**, 84–93

30. A.G. Lindgren, K.F. Gong (1978), Position and velocity estimation via bearing observations, *IEEE Trans. Aero. Elect. Syst.*, **14**, 564–577

31. P.J. Galkowski, M.A. Islam (1991), An alternative derivation of the modified gain function of Song and Speyer, *IEEE Trans. Automatic Control*, **36**, 1323–1326

32. J. Lévine, R. Marino (1992), Constant-speed target tracking via bearings-only measurements, *IEEE Trans. Aero. Elect. Syst.*, **28**, 174–182

33. M. Wax (1983), *Position location from sensors with position uncertainty*, IEEE Trans. Aero. Elect. Syst., **19**, 658–662

34. Appendix E, labelled "Emitter location accuracy" of a handwritten report held in DSTO's Electronic Warfare and Radar Division. Its origin is unknown but it appears to be about ten years old.

Appendix A Posterior CRLB

The posterior Cramér–Rao lower bound was discussed in Section 3.2.1, and the ideas underlying it are outlined in this appendix.

When using a recursive geolocation routine, we need to take into account the prior knowledge available at each step of the recursion. The state $\boldsymbol{x}$ that we seek now takes on a new value as each measurement arrives. Hence, index it with the measurement number: the measurement set $Z_k \equiv \{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_k\}$ estimates the latest value of the state, $\boldsymbol{x}_k$. Prior information is given by the quantity $\boldsymbol{z}_0$, which will be a measurement, but is not counted as part of the data set.

Instead of dealing with the conditional density $p(Z_k|\boldsymbol{x}_k)$ as in Section 3.2.1, the state $\boldsymbol{x}_k$ is now treated as a random variable in its own right, and so the combined density $p(Z_k, \boldsymbol{x}_k)$ is used to compute the bayesian Fisher information at each time k:

$$J_B(k) \equiv \mathbb{E}_{Z_k,\boldsymbol{x}_k}\left\{\nabla_{\boldsymbol{x}_k}\left(\nabla^t_{\boldsymbol{x}_k} L(Z_k, \boldsymbol{x}_k)\right)\right\}, \tag{A1}$$

where $\mathbb{E}_{Z_k,\boldsymbol{x}_k}\{\cdot\} \equiv \mathbb{E}_{Z_k}\left\{\mathbb{E}_{\boldsymbol{x}_k}\{\cdot\}\right\}$. Define $L(\boldsymbol{x}_k)$ via

$$\begin{aligned} L(Z_k, \boldsymbol{x}_k) &= -\ln p(Z_k, \boldsymbol{x}_k) \\ &= -\ln p(\boldsymbol{x}_k) - \ln p(Z_k|\boldsymbol{x}_k) \\ &\equiv L(\boldsymbol{x}_k) + L(Z_k|\boldsymbol{x}_k)\,. \end{aligned} \tag{A2}$$

Equations (A1) and (A2) combine to give

$$J_B(k) = \underbrace{\mathbb{E}_{\boldsymbol{x}_k}\left\{\nabla_{\boldsymbol{x}_k}\left(\nabla^t_{\boldsymbol{x}_k} L(\boldsymbol{x}_k)\right)\right\}}_{\equiv J_P(k),\text{ the prior information}} + \underbrace{\mathbb{E}_{Z_k,\boldsymbol{x}_k}\left\{\nabla_{\boldsymbol{x}_k}\left(\nabla^t_{\boldsymbol{x}_k} L(Z_k|\boldsymbol{x}_k)\right)\right\}}_{= \mathbb{E}_{\boldsymbol{x}_k}\{J_{NB}(k)\},\text{ the data information}}, \tag{A3}$$

where we have replaced $\mathbb{E}_{Z_k,\boldsymbol{x}_k}$ by $\mathbb{E}_{\boldsymbol{x}_k}$ in the first term on the right in (A3), since its argument does not depend on the measurements Z_k. Also, as is typically done to simplify calculations in bayesian theory, the second term of (A3), $\mathbb{E}_{\boldsymbol{x}_k}\{J_{NB}(k)\}$, is usually replaced by $J_{NB}(k)$. This corresponds to using an unchanging set of initial conditions in a set of Monte Carlo simulations.

Suppose that an initial measurement $\boldsymbol{z}_0$ (i.e. prior information) has indicated a target state $\boldsymbol{x}_0$, and assume $\boldsymbol{x}_0$ is normally distributed: $p(\boldsymbol{x}_0) \sim \mathcal{N}(\widehat{\boldsymbol{x}}_0, P_0)$, by which is meant

$$p(\boldsymbol{x}_0) = \frac{1}{\sqrt{|2\pi P_0|}} \exp \frac{-1}{2}\left(\boldsymbol{x}_0 - \widehat{\boldsymbol{x}}_0\right)^t P_0^{-1}\left(\boldsymbol{x}_0 - \widehat{\boldsymbol{x}}_0\right)\,. \tag{A4}$$

Then if $\boldsymbol{u}_k$ and $\boldsymbol{v}_k$ are set to zero in (3.46) (as done in bearings-only tracking), it follows that $\boldsymbol{x}_k$ is normally distributed: $p(\boldsymbol{x}_k) \sim \mathcal{N}(\widehat{\boldsymbol{x}}_k, P_k)$. Setting $F_k \equiv F$, a constant matrix, the relevant expressions now involve the k^{th} power of F:

$$\widehat{\boldsymbol{x}}_k = F^k \widehat{\boldsymbol{x}}_0\,, \quad P_k = F^k P_0 (F^k)^t. \tag{A5}$$

In that case, the prior information turns out to be

$$J_P(k) = P_k^{-1}. \tag{A6}$$

Thus, the calculation of the posterior CRLB in (A3) reduces to a calculation of J_{NB}, the non-bayesian Fisher information. The calculation of this matrix is somewhat laborious; the remainder of this appendix outlines the way it is done, for an emitter modelled as having constant velocity.

For this constant velocity case, the emitter state is defined in terms of its position (x_k, y_k) and velocity $(\dot{x}_k, \dot{y}_k)$ at time k:

$$\boldsymbol{x}_k = \begin{bmatrix} x_k & y_k & \dot{x}_k & \dot{y}_k \end{bmatrix}^t . \tag{A7}$$

Beginning with (3.3, 3.6), we need the conditional probability

$$p(Z_k|\boldsymbol{x}_k) \equiv \frac{1}{\sqrt{2\pi R}} \exp\left[\frac{-1}{2R}\sum_{i=1}^{k} [z_i - h(\boldsymbol{x}_k, i)]^2\right], \tag{A8}$$

which directly gives $L(Z_k|\boldsymbol{x}_k)$. The quantity $h(\boldsymbol{x}_k, i)$ is the same as used in (3.52), being a sort of backwards prediction of the exact bearing that should be seen at time $i \leqslant k$, given the current time k and current state $\boldsymbol{x}_k$. With equal time steps Δt, we can write the emitter's position at some intermediate time i as:

$$(x_i, y_i) = (x_k, y_k) + (\dot{x}_k, \dot{y}_k)(i-k)\Delta t\,, \tag{A9}$$

so that (omitting the quadrant-dependent constant in the following inverse tan functions, which will differentiate to zero anyway—see the comment on page 19)

$$\begin{aligned} h(\boldsymbol{x}_k, i) &\equiv \tan^{-1}\frac{y_i - r_{i,y}}{x_i - r_{i,x}} \equiv \tan^{-1}\frac{\Delta y_i}{\Delta x_i} \quad \text{(for all emitter models)} \\ &= \tan^{-1}\frac{y_k + \dot{y}_k(i-k)\Delta t - r_{i,y}}{x_k + \dot{x}_k(i-k)\Delta t - r_{i,x}} \quad \text{(constant velocity model only),} \end{aligned} \tag{A10}$$

where $(r_{i,x}, r_{i,y})$ is the receiver position at time i. The Fisher information matrix has the raw form of

$$J_{NB}(k) = \mathbb{E}_{Z_k}\left\{\nabla_{\boldsymbol{x}_k}\left(\nabla^t_{\boldsymbol{x}_k} L(Z_k|\boldsymbol{x}_k)\right)\right\}. \tag{A11}$$

For convenience, if we rewrite the state as

$$\boldsymbol{x}_k = \begin{bmatrix} x_k^1 & x_k^2 & x_k^3 & x_k^4 \end{bmatrix}^t, \tag{A12}$$

then, with further effort, the Fisher information matrix can be shown to have as its $(m,n)^{\text{th}}$ element:

$$J_{NB}(k)\Big|_{mn} = \frac{1}{R}\sum_i \frac{\partial h(\boldsymbol{x}_k, i)}{\partial x_k^m}\,\frac{\partial h(\boldsymbol{x}_k, i)}{\partial x_k^n}\,. \tag{A13}$$

Actually this last expression holds for emitter models with arbitrary motion, not just constant velocity; what does change for different emitter models is the form of $h(\boldsymbol{x}_k, i)$ in the second line of (A10), where the numerator and denominator would contain extra Taylor series terms involving higher derivatives of position.

For the constant velocity case, further manipulation leads to

$$J_{NB}(k) = \frac{1}{R}\sum_{i=1}^{k}\frac{1}{\left(\Delta x_i^2+\Delta y_i^2\right)^2}\times \begin{bmatrix} \Delta y_i^2 & -\Delta y_i \Delta x_i & \Delta y_i^2 (i-k)\Delta t & -\Delta y_i \Delta x_i (i-k)\Delta t \\ & \Delta x_i^2 & -\Delta y_i \Delta x_i (i-k)\Delta t & \Delta x_i^2 (i-k)\Delta t \\ & & \Delta y_i^2 (i-k)^2 \Delta t^2 & -\Delta y_i \Delta x_i (i-k)^2 \Delta t^2 \\ & \text{(symmetric)} & & \Delta x_i^2 (i-k)^2 \Delta t^2 \end{bmatrix}. \tag{A14}$$

The bayesian Fisher information $J_B(k)$ then follows from (A3, A6, A14).

Appendix B Total Least Squares

Here we supply the relevant details referred to in Section 3.6.1.

When the matrix H of (3.14) itself contains noise, the usual pseudo inverse is not guaranteed to give the most accurate estimate of the state $\boldsymbol{x}$. In this case the method of *Total Least Squares* can sometimes give better results. To construct the method, begin by rewriting (3.14) as

$$\begin{bmatrix} H & -\boldsymbol{z} \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ 1 \end{bmatrix} = \text{residual.} \tag{B1}$$

Suppose we multiply both sides by $\begin{bmatrix} H & -\boldsymbol{z} \end{bmatrix}^t$:

$$\begin{bmatrix} H & -\boldsymbol{z} \end{bmatrix}^t \begin{bmatrix} H & -\boldsymbol{z} \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ 1 \end{bmatrix} = \begin{bmatrix} H & -\boldsymbol{z} \end{bmatrix}^t \text{residual.} \tag{B2}$$

In a noiseless world the right hand side would be zero, in which case $A \equiv \begin{bmatrix} H & -\boldsymbol{z} \end{bmatrix}^t \begin{bmatrix} H & -\boldsymbol{z} \end{bmatrix}$ (a square matrix) would be singular. In practice we will perturb it to produce a new singular version, in the least perturbative way, by zeroing its smallest eigenvalue. This can be done through first finding three matrices U, S, V that make up the singular value decomposition of $\begin{bmatrix} H & -\boldsymbol{z} \end{bmatrix}$:

$$\begin{bmatrix} H & -\boldsymbol{z} \end{bmatrix} = USV^t$$

Matlab: `[U, S, V] = svd([H, -z], 0).` (B3)

The matrices U, S and V have useful properties that need not concern us here, except that S is diagonal, having as its entries the square roots of the eigenvalues of A. If we arrange these in descending order down the diagonal,[3] then we can perturb A to be singular by zeroing the last eigenvalue of S, to produce a new matrix $\bar{S}$. In that case, the new matrix $U\bar{S}V^t$ defines new H and z matrices:

$$\begin{bmatrix} \bar{H} & -\bar{\boldsymbol{z}} \end{bmatrix} \equiv U\bar{S}V^t \,. \tag{B4}$$

Solving the new system

$$\begin{bmatrix} \bar{H} & -\bar{\boldsymbol{z}} \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ 1 \end{bmatrix} = 0 \,, \quad \text{i.e.} \quad \bar{\boldsymbol{z}} = \bar{H}\boldsymbol{x} \,, \tag{B5}$$

produces a possibly better estimate of the state $\boldsymbol{x}$. In practice, the calculation is further simplified because the equation which is to be solved,

$$U\bar{S}V^t \begin{bmatrix} \boldsymbol{x} \\ 1 \end{bmatrix} = 0 \,, \tag{B6}$$

can be rewritten in terms of the columns of the $n \times n$ matrix V. If the state $\boldsymbol{x}$ has n entries, then

$$\text{for each row } i, \quad \begin{bmatrix} \boldsymbol{x} \\ 1 \end{bmatrix} \cdot \sum_{j=1}^{n-1} U_{ij} S_{jj} V_{\text{column } j} = 0 \,, \tag{B7}$$

[3] U and V would have to be altered to suit, but this preferred arrangement is automatically produced by Matlab anyway. We need do nothing extra.

so that $\begin{bmatrix}\boldsymbol{x}\\1\end{bmatrix}$ must be orthogonal to the first $n-1$ columns of V. But a special property of V is that it is orthogonal: its columns are all mutually orthonormal. Hence $\begin{bmatrix}\boldsymbol{x}\\1\end{bmatrix}$ must be proportional to the last column of V. In that case, the total least squares estimate of $\boldsymbol{x}$ is just

$$\bar{\boldsymbol{x}} = \frac{1}{V_{nn}}\begin{bmatrix}V_{1n}\\ \vdots \\ V_{n-1,n}\end{bmatrix}. \tag{B8}$$

This is all easily programmed in Matlab, especially since the matrices U, S are not used. In summary, when solving for $\boldsymbol{x}$ with a noisy H:

$$H\boldsymbol{x} = \boldsymbol{z} + \text{noise}, \tag{B9}$$

the two approaches of least squares and total least squares can be programmed in the following way:

Least squares:

$$\bar{\boldsymbol{x}} = H^{\#}\boldsymbol{z}$$

Matlab:
```
x = pinv(H) * z
```
(B10)

Total least squares:

$$USV^t = \begin{bmatrix}H & -\boldsymbol{z}\end{bmatrix} \text{ (s.v.d.)}; \quad \bar{\boldsymbol{x}} = \frac{1}{V_{nn}}\begin{bmatrix}V_{1n}\\ \vdots \\ V_{n-1,n}\end{bmatrix}$$

Matlab:
```
[notNeeded, notNeeded, V] = svd([H, -z], 0);
x = V(1:end-1, end)/V(end, end)
```
(B11)

Appendix C Running the Matlab Geolocation Simulator

The Matlab programme referred to in this report is used to investigate the tracking algorithms of Sections 3.7–3.10. It allows us to specify fairly general flight paths for one receiver and one emitter. It then generates noisy bearings at regular intervals along the flight paths, and uses our choice of either the CPLE, Gauss–Newton, RLS, or either type of Kalman filter to estimate the emitter's flight path. It also plots Cramér–Rao uncertainty ellipses as described in Section 5.1 and [4], that serve as a guide to how accurately we can ever hope to locate the emitter.

Here we describe two possible ways of running the Matlab gui.

1. To run the gui in its simplest form, type `geolocationGui` at the Matlab command prompt, which will run the `geolocationGui.m` file. This brings up the gui. Parameters can then be entered into various boxes. Pressing the appropriate button draws and processes the flight paths.

2. If a better feel for the statistics of the final estimation is needed, a Monte Carlo approach can be used. This runs the relevant routine many times for various emitter-receiver geometries. Instead of typing `geolocationGui` from the command line, type `manyRuns` instead, which runs the code in `manyRuns.m`. The gui will start as usual, at which point the appropriate parameters must be entered into it; this is just equivalent to running `geolocationGui.m` for each set of parameters and entering these by hand into the gui. Currently `manyRuns.m` is only set up to handle a stationary emitter and constant velocity receiver.

Running `geolocationGui.m` (the Gui)

After typing `geolocationGui`, the window in Figure C1 appears. The plot area for flight paths is a flat area with north upwards and east to the right. It starts off with a nominal axes scaling, although the scaling can be changed at any time by clicking the Change axes limits button.

Once appropriate axes limits have been chosen, the paths of both emitter and receiver are plotted on this map. They can be plotted in either order, and are both recorded in the same way. For example, we enter emitter waypoints by first clicking on the button labelled Start recording emitter waypoints, then clicking the left mouse button to place each waypoint on the map. We finish by clicking the same gui button, now relabelled Stop recording emitter waypoints.

The actual flight speeds and bearing measurement times involved are changed by entering numbers in the two input boxes: `Time between receiver waypoints` and `Time between bearing measurements`. The time between emitter waypoints cannot be set, since it is determined by the number of waypoints for emitter and receiver we input, together with the time between receiver waypoints.

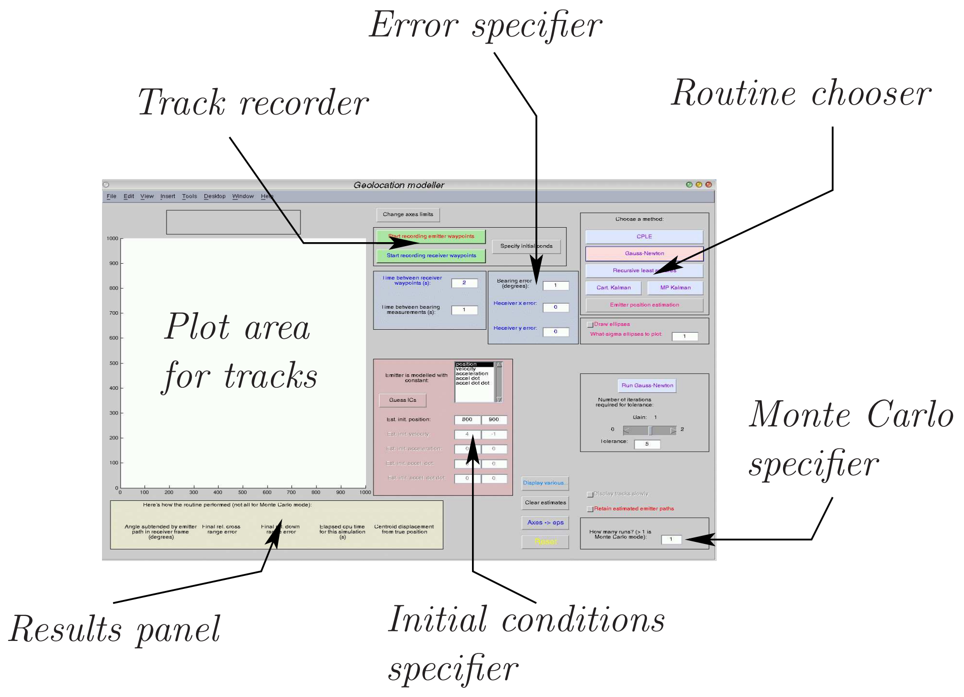

Figure C1: *Startup window for the geolocation gui*

As an alternative to specifying the flight paths this way, we can use an initial condition approach. First click on the Specify initial conds button. This brings up a new set of buttons and input boxes. Before doing anything else, ensure that the time between bearing measurements has been entered. Then, for each of the receiver and emitter (in either order), enter the flight time (which should be the same for both), and the initial position, velocity, etc. Clicking on Draw emitter path or Draw receiver path then plots a path based on these initial conditions.

Next specify how many runs we wish to make. Specifying one run makes the receiver and emitter fly their paths just once; the receiver takes noisy bearing measurements at intervals given by the `Time between bearing measurements` input box, and an answer for the estimated position or track of the emitter is computed. If a number greater than one is chosen, this process is repeated many times, for a new set of noise values each time. Each estimated emitter track is overlaid on the screen at the end of the computation—or whenever Matlab decides to draw its graphics buffer to the screen, which currently is always at the end of the whole set of computations.

Checking the Retain estimated emitter paths box means that no matter how many runs are made, successive paths will remain on the screen. To clear these runs, click the Clear estimates button. This prevents Matlab from doing a slow screen refresh with these paths in place as part of its screen management before it starts the next run.

Now enter the bearing error into the `Bearing error (degrees)` box. (Note that all of the input boxes are independent of each other, so the various inputs can be entered in any order.) This is a value that will become the standard deviation of a normal distribution centred about 0°, from which Matlab will draw random values to add to the exact, true, bearing. This noisy bearing is what Matlab then actually uses to perform its geolocation computation.

We can also enter receiver position errors into the `Receiver x error` and `Receiver y error` boxes. These numbers, analogously to the bearing error, are the standard deviations of normal distributions about zero from which will be randomly drawn numbers to add to the receiver x and y positions. They reflect our uncertainty of where the receiver is, due to INS and GPS errors. These numbers are ordinarily quite small, if they are present at all.

A geolocation method must be selected from the following:

CPLE. To run this routine first click the CPLE button, which causes a new set of buttons to be drawn. There is just one button to press to run the routine: (another) CPLE, since this routine requires no seed estimate. Pressing this button causes Matlab to do the following with the flight paths. First it interpolates the waypoints so as to make the same number of points in each track, spaced in time by the bearing measurement interval. That way the new waypoints can be paired off, so that the emitter position corresponding to each receiver measurement is unambiguous. Then a cubic spline curve is computed through each set of points to make two smooth paths.

Next, Matlab builds a list of noisy bearings, one for each receiver-emitter waypoint pair. Once this is done, further processing of the noisy bearings depends on the routine.

If we have modelled the emitter's motion as constant position, then the estimate produced by the CPLE routine will be a single point. Likewise if we have chosen constant velocity as the model, the routine works out an initial position and initial velocity, and plots the constant velocity path resulting from these, which is necessarily a straight line. (This fact should be noted because the other routines might do this differently; e.g. the Kalman routine tracks the emitter, and will seldom—essentially never—produce a straight line for the same constant velocity emitter.) Similarly, a set of initial conditions is produced and used to plot the emitter motion, when we model that motion as having higher derivatives.

After running the CPLE routine, its results can, if we choose, be used as initial conditions of any of the other routines, which all require a seed. Do this by clicking the Use result to set new ICs button.

Gauss–Newton. Choose this with the Gauss–Newton button. Enter a gain and tolerance in the box that appears. The gain is a factor around unity that multiplies the stepping size calculated by Gauss–Newton, in the hope of helping the iterations converge faster. The tolerance is the distance on the plot area such that once the change in (final) position from one iteration to the next is less than this amount, the routine will terminate.

Next, estimate the initial conditions. They might have been already estimated by a preliminary run of the CPLE; but if not, choose an emitter motion model from the list next to `Emitter is modelled with constant:`. It is possible to "cheat" by hitting the Guess ICs button. This calculates the actual initial conditions for the emitter path that was entered, and is a way of entering something at least half way reasonable, as a way of experimenting with different scenarios for which guessing reasonable initial conditions is difficult.

Now click the newly appeared Gauss–Newton button to run the routine. Its iterations stop once the tolerance has been reached. Iterations also stop once 25 of them have occurred, since this indicates that the routine is probably not converging and should be abandoned. (This number 25 is just the value of the `maxIterations` variable in `Gauss--Newton.m` and can be changed there if required.)

As with the CPLE routine, Gauss–Newton calculates a set of initial conditions for the emitter. Thus if the emitter is modelled with constant velocity, an absolutely straight line will necessarily be drawn.

Recursive Least Squares. Clicking Recursive least squares prints the relevant buttons to the gui. Initial conditions and a model of the emitter's motion must be specified.

There are now two parameters to specify to run RLS: `Lambda` and `Q`, as defined in Section 3.9. Typical values might be `Lambda = 1` and `Q = 1e8` (i.e. 10^8). Pressing the new Recursive least squares button runs the routine.

The type of track to expect of the estimated emitter plot is different for this and the next techniques. For example, if we choose the emitter as being modelled with constant position, then unlike the CPLE and G–N case, the resulting plot will almost certainly not be just one point. The reason is that RLS and the Kalman filters are not batch processors, and so do not produce just one state estimate from their processing. Rather, they plot a point whose location is then updated as each new bearing arrives. Similarly if we choose to model the velocity as constant: a straight line is not expected to be output, since as before, the position of the emitter is simply being estimated and updated for each bearing.

Cartesian Extended Kalman. Click on the Cart. Kalman button to run this. Besides our supplying initial conditions as before, the routine also requires the parameters labelled `P_0|0` and `Q` as defined in Section 3.10. Typical values might be 1000 and 1. Then click the Cartesian Kalman button.

Modified Polar Extended Kalman. For this, press the MP Kalman button and enter values for `sigma_r`, `sigma_v` and `Approx. range`. These are, respectively, the errors in the initial position and speed, and an estimate of the range (some intermediate value can be given if the range changes greatly for the supplied paths). Then click on the Modified Polar Kalman button.

Other Buttons and Output Fields

Display various... There is sometimes a need to produce various numbers indicating the performance of an algorithm. Clicking on this button writes these numbers to the Matlab command line. It will produce an error message when clicked if some of the information coded to be output doesn't exist.

Axes → eps Clicking this produces an encapsulated postscript file of the current plot. The allocated filename is time-tagged to prevent accidental overwriting.

Reset This clears all tracks from the gui and sets various other quantities to sensible values. Use this when a new set of flight paths is required to be drawn. If we just add more waypoints to a plot without hitting Reset first, they will be added to the existing flight paths.

Emitter position estimation This button implements the work of Kim Brown as outlined in Section 3.2. Clicking on it brings up a separate window for parameter input. The scenario it is considering, described comprehensively in [4], is that of a plane flying a great circle over Earth, taking bearings of a stationary emitter. The numbers to be input relate to Figure C2:

`Broadside angle (degrees)` As in Figure C2.

`Emitter range (km)` The units of km are purely arbitrary.

`Bearing error (degrees)`

`Number of points` This is the number of points along the receiver flight path, at each of which the receiver takes a bearing measurement.

`Receiver speed (km/h)`

`Time (s) between bearing measurements` The unit of seconds is arbitrary. The length of the receiver path flown depends on this time and the number of points specified along the path.

Clicking the Emitter position estimation button calculates the Cramér–Rao lower bound in such a way as to draw an uncertainty ellipse, which is really meant to be superimposed on the emitter position. This is a measure of the most optimistic error we can hope for in the specified situation.

This emitter position estimation theory has also been implemented in the main geolocation routines, provided we are modelling the emitter as stationary there. If we are, then the same theory is used to draw a Cramér–Rao error ellipse, this time superimposed on the actual position of the emitter. This is useful, because it allows us to compare the geolocated position with the Cramér–Rao error ellipse. However, note that the Cramér–Rao theory applies to unbiased techniques, and the techniques used in the gui are not necessarily unbiased—as is demonstrated whenever a spread of Monte Carlo emitter estimates does not cluster about the true emitter position.

Plotting the ellipse is governed by an edit field in the main gui: `What-sigma ellipses to plot`. If set to e.g. 2, then the ellipse resulting is really a contour of the probability distribution at two standard deviations. For accurate geolocation scenarios, the 1σ ellipse is usually too small to see, so raising the value of σ is equivalent to magnifying the ellipse to make it visible. The Draw ellipses button must be checked to switch this facility on, since plotting the ellipses can be computationally expensive.

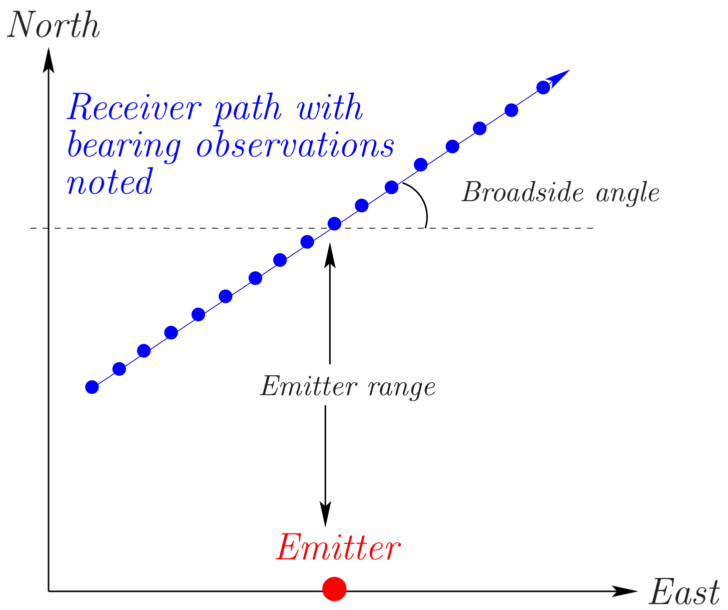

Figure C2: *Emitter position estimation scenario*

Display tracks slowly Checking this button draws the tracks point by point, as a visual aid only. It's only enabled for the recursive routines, to emphasise the fact that these routines operate by updating the emitter estimate as each bearing-capture point of the flight path is reached. Depending on which routine has been chosen, only certain combinations of this button together with the Draw ellipses button are enabled. This is a coding choice, due to the practicalities of calculating the posterior Cramér–Rao bound for the recursive routines.

`How many runs?` If we want to do a Monte Carlo run, enter a number greater than 1 here. In principle, after each run the resulting estimate of the emitter path will be drawn to the screen, as well as the current run number being printed on the Matlab command line; but in practice, Matlab only draws its graphical buffer at the end of the entire Monte Carlo set. Next to this box, two values for the circular error probable will be printed, for 50% and 95%. That is, the 50% number is the distance out from the emitter's final position that we must go in order to encompass 50% of the estimates to that position. Similarly for the 95% number.

`Here's how the routine performed` After every run, several performance indicators are printed into this rectangle in the gui. These are:

`Angle subtended by the emitter path in receiver frame (degrees)` This only has real use if the receiver is not following an overly convoluted path. The label is worded in this way to take into account the fact that the emitter might be moving. In the simplest case of a stationary emitter, this angle is identical to the angle subtended by the receiver path at the emitter.

`Final rel. cross-range error` Refer to Figure 2 for the definition of cross-range error. The *relative* cross-range error is the cross-range error divided by the range. This quantity is really the answer to the more specific question: at the final position of the receiver, what is the relative cross-range error in the final estimated position of the emitter?

`Final rel. down-range error` This is analogous to the relative cross-range error.

`Elapsed cpu time for this simulation (s)` The elapsed cpu time in seconds is a good parameter to compare the speed of the different routines.

`Centroid displacement from true position` Whether we are running in Monte Carlo mode (in which case a set of emitter paths will be plotted), or not (in which case just one emitter path will be plotted), there will be one or more estimates for the final emitter position. This set has a centroid. The displacement of this centroid from the true emitter position is a good indicator of how well the geolocation routine is performing.

In the case of running Gauss–Newton, the centroid placement might be cause for confusion in that one might ask: why is the emitter placement error (output after the calculation) sometimes greater than the tolerance that we have specified? This is because: in specifying the tolerance, we are telling the algorithm to stop when the distance between successive estimates to the emitter position becomes smaller than the tolerance, so that it doesn't keep iterating unnecessarily for no real improvement in its estimate. On the other hand, the emitter placement error is the distance between the last point and the actual emitter position. It might be that G–N has done a bad job, and converged on a point far from the actual emitter position. In that case the emitter placement error will be large, even though the iteration steps have reduced to the step size required to stop the algorithm.

Running `manyRuns.m` (Batch Mode)

The file `manyRuns.m` contains code that runs a batch of geolocation scenarios. The basic layout is covered in Section 6.2, and shown in Figure 12 on page 46. The emitter is at rest on the y-axis, while the emitter has a constant velocity at some angle α to the x-axis. It sweeps out a path of angle θ_s as seen by the emitter, in a time Δt, and it makes bearing observations at the rate of one per second. The code is run by typing `manyRuns` at the Matlab prompt, at which we then supply values for the various parameters used to control the runs, as arrays. These are the bearing error, θ_s, α, and Δt. So to enter an array of values $0, 5, 10, 15, \ldots, 80$, we just enter `0:5:80`, while to enter an array with a changing increment, we enter the array explicitly: `[2 3 5 7 11]`

The number of Monte Carlo runs and the routine used must also be specified at the prompt. All runs, apart from when using the CPLE routine, are automatically seeded by first running CPLE (which needs no initial conditions). The chosen routine is then run as many times as entered for the Monte Carlo number.

`manyRuns.m` produces two files. The main one is a `.mat` file whose name we supply when prompted by `manyRuns.m`. This name is time-tagged to prevent any accidental write-over: if we specify the name `fred` at time 2:02:41 pm, the actual name will be

```
fred_23_Mar_2005_14_02_41.mat
```

The second file produced is a "diary file" `fred_23_Mar_2005_14_02_41.diary` containing a history of the runs, in case this is needed for checking in the event of a computer crash; this can be deleted when `manyRuns.m` finishes.

The `.mat` file is then read by `PlotMatrixCurves.m`, which reads its data and plots various graphs. (`PlotMatrixCombined.m` also does a similar job, but is a little less evolved. It puts all of its plots onto one page, in the event that they are to be included this way in a report.) `PlotMatrixCurves.m` asks for several inputs:

Name of input file. That is, the above `.mat` file containing the plot data.

What to plot on y-axis. Refer to Section 6.2. Choices are relative down- and cross-range errors, and circular error probable. Choosing e.g. relative down-range error, the relevant choice is presented as "`Relative down-range error and sigma_R/R`". What this indicates is that the relative down-range error (the statistical error output from `geolocationGui.m` run one or more times through `manyRuns.m`) will be plotted on the same graph as the "rule of thumb error" from (6.1).

Bearing error values. Here we are presented with a range of bearings (in degrees) that have been used to create the data now held in the `.mat` file. For example, a line saying `0:0.1:0.5` means that all the values 0, 0.1, 0.2, ..., 0.5 have been used as bearing errors. Either enter one of these, or enter nothing at all (just hit return). Entering a specific number means we wish to plot a slice through the matrix of data at this value of bearing error. If nothing is entered, all of the bearing errors will become the x-axis. Note that if only ever one bearing error was already stipulated in `manyRuns.m`, then we will not be asked to specify that bearing again.

theta_s values. Similarly we enter a value for the swept angle θ_s in degrees. Note that if we enter nothing here as well as for the bearing error, `PlotMatrixCurves.m` will crash as it tries to put two sets of values along the x-axis. (This nonrobustness is not crucial.) As above, if only one θ_s value was originally stipulated, then we are not asked for it again here.

alpha. We also enter a value for α (degrees) if it's asked for (as for bearing and θ_s).

delta_tReceiver values. Here, if asked, we enter the time Δt taken to fly the receiver flight path, in seconds. Bearing measurements will be taken at one second intervals.

After entering these values the relevant graph is plotted. We can also plot relative down- and cross-range errors and CEPs as functions of two variables, using `PlotMatrixSurfaces.m`.

DISTRIBUTION LIST

Numerical Calculations for Passive Geolocation Scenarios

Don Koks

	Number of Copies
DEFENCE ORGANISATION	
Task Sponsor	
Director General Aerospace Development	1 (printed)
S&T Program	
Chief Defence Scientist	1
Deputy Chief Defence Scientist Policy	1
AS Science Corporate Management	1
Director General Science Policy Development	1
Counsellor, Defence Science, London	Doc. Data Sheet
Counsellor, Defence Science, Washington	Doc. Data Sheet
Scientific Adviser to MRDC, Thailand	Doc. Data Sheet
Scientific Adviser Joint	1
Navy Scientific Adviser	Doc. Data Sheet and Dist. List
Scientific Adviser, Army	Doc. Data Sheet and Dist. List
Air Force Scientific Adviser	1
Scientific Adviser to the DMO	Doc. Data Sheet and Dist. List
Deputy Chief Defence Scientist Platform and Human Systems	Doc. Data Sheet and Exec. Sum.
Len Sciacca, CEWRD	Doc. Data Sheet and Dist. List
Alasdair M^{c}Innes, RL EWS, EWRD	Doc. Data Sheet and Dist. List
Simon Rockliff, Head AS group, EWRD	1 (printed)
Anthony Schellhase, Head MS group, EWRD	Doc. Data Sheet
Sanjeev Arulampalam, MOD	1 (printed)
Mark Rutten, ISRD	1 (printed)
Don Koks, EWRD	5 (printed)
DSTO Library and Archives	
Library, Edinburgh	2 (printed)
Defence Archives	1
Capability Development Group	
Director General Maritime Development	Doc. Data Sheet

Director General Capability and Plans	Doc. Data Sheet
Assistant Secretary Investment Analysis	Doc. Data Sheet
Director Capability Plans and Programming	Doc. Data Sheet
Director General Australian Defence Simulation Office	Doc. Data Sheet
Chief Information Officer Group	
Head Information Capability Management Division	Doc. Data Sheet
AS Information Strategy and Futures	Doc. Data Sheet
Director General Information Services	Doc. Data Sheet
Strategy Group	
Assistant Secretary Strategic Planning	Doc. Data Sheet
Assistant Secretary Governance and Counter-Proliferation	Doc. Data Sheet
Navy	
Deputy Director (Operations) Maritime Operational Analysis Centre, Building 89/90, Garden Island, Sydney	Doc. Data Sheet and Dist. List
Deputy Director (Analysis) Maritime Operational Analysis Centre, Building 89/90, Garden Island, Sydney	
Director General Navy Capability, Performance and Plans, Navy Headquarters	Doc. Data Sheet
Director General Navy Strategic Policy and Futures, Navy Headquarters	Doc. Data Sheet
Air Force	
SO (Science), Headquarters Air Combat Group, RAAF Base, Williamtown	Doc. Data Sheet and Exec. Sum.
Army	
ABCA National Standardisation Officer, Land Warfare Development Sector, Puckapunyal	Doc. Data Sheet (pdf format)
J86 (TCS Group), DJFHQ(L), Enoggera QLD	Doc. Data Sheet
SO (Science), Land Headquarters (LHQ), Victoria Barracks, NSW	Doc. Data Sheet and Exec. Sum.
SO (Science), Special Operations Command (SOCOMD), R5-SB-15, Russell Offices Canberra	Doc. Data Sheet and Exec. Sum.
SO (Science), DJFHQ(L), Enoggera QLD	Doc. Data Sheet
Joint Operations Command	
Director General Joint Operations	Doc. Data Sheet
Chief of Staff Headquarters Joint Operation Command	Doc. Data Sheet
Commandant, ADF Warfare Centre	Doc. Data Sheet
Director General Strategic Logistics	Doc. Data Sheet
COS Australian Defence College	Doc. Data Sheet

Intelligence and Security Group	
Assistant Secretary, Concepts, Capabilities and Resources	1
DGSTA, DIO	1
Manager, Information Centre, DIO	1
Director Advanced Capabilities, DIGO	Doc. Data Sheet
Defence Materiel Organisation	
Deputy CEO, DMO	Doc. Data Sheet
Head Aerospace Systems Division	Doc. Data Sheet
Head Maritime Systems Division	Doc. Data Sheet
Program Manager Air Warfare Destroyer	Doc. Data Sheet
Guided Weapon and Explosive Ordnance Branch GWEO	Doc. Data Sheet
CDR Joint Logistics Command	Doc. Data Sheet

OTHER ORGANISATIONS

National Library of Australia	1
NASA (Canberra)	1

UNIVERSITIES AND COLLEGES

Australian Defence Force Academy Library	1
Head of Aerospace and Mechanical Engineering, ADFA	1
Hargrave Library, Monash University	Doc. Data Sheet

INTERNATIONAL DEFENCE INFORMATION CENTRES

US: Defense Technical Information Center	1
UK: Dstl Knowledge Services	1
Canada: Defence Research Directorate R&D Knowledge and Information Management (DRDKIM)	1
NZ: Defence Information Centre	1

ABSTRACTING AND INFORMATION ORGANISATIONS

Library, Chemical Abstracts Reference Service	1
Engineering Societies Library, USA	1
Materials Information, Cambridge Scientific Abstracts, USA	1
Documents Librarian, The Center for Research Libraries, USA	1

INFORMATION EXCHANGE AGREEMENT PARTNERS

National Aerospace Laboratory, Japan	1
National Aerospace Laboratory, Netherlands	1

SPARES

DSTO Edinburgh Library	5 (printed)

Total number of copies: 40; printed 16, pdf 24.

Page classification: UNCLASSIFIED

DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION DOCUMENT CONTROL DATA	1. CAVEAT/PRIVACY MARKING

2. TITLE Numerical Calculations for Passive Geolocation Scenarios	3. SECURITY CLASSIFICATION Document (U) Title (U) Abstract (U)
4. AUTHOR Don Koks	5. CORPORATE AUTHOR Defence Science and Technology Organisation PO Box 1500 Edinburgh, SA 5111, Australia

6a. DSTO NUMBER DSTO–RR–0319	6b. AR NUMBER AR–013-779	6c. TYPE OF REPORT Research Report	7. DOCUMENT DATE January, 2007

8. FILE NUMBER E9505–25–199	9. TASK NUMBER AIR 00/069	10. SPONSOR DGAD	11. No. OF PAGES 64	12. No OF REFS 34

13. URL OF ELECTRONIC VERSION http://www.dsto.defence.gov.au/corporate/reports/DSTO–RR–0319.pdf	14. RELEASE AUTHORITY Chief, Electronic Warfare and Radar Division

15. SECONDARY RELEASE STATEMENT OF THIS DOCUMENT

Approved for Public Release

OVERSEAS ENQUIRIES OUTSIDE STATED LIMITATIONS SHOULD BE REFERRED THROUGH DOCUMENT EXCHANGE, PO BOX 1500, EDINBURGH, SA 5111

16. DELIBERATE ANNOUNCEMENT

No Limitations

17. CITATION IN OTHER DOCUMENTS

No Limitations

18. DSTO RESEARCH LIBRARY THESAURUS

geolocation	tracking
Cramér–Rao bounds	Kalman filters
least-squares method	maximum-likelihood estimation
circular error probable	time difference of arrival
Gauss–Newton	

19. ABSTRACT

This report reviews work done in gaining some familiarity with methods of passive geolocation, and a search for rules of thumb that might tell us how to optimise the geolocation for a given scenario. We first cover the main approaches to collecting angle of arrival data and point out typical accuracies. Following this is an account of the mathematics used to analyse this data to produce an estimate of an emitter's location. We then give an overview of some of the literature, and finish by demonstrating a Matlab programme that runs several geolocation algorithms. Simple rules of thumb that specify how to fly a baseline and take data, so as to maximise the accuracies of the different techniques, do not appear to be readily derivable.

Page classification: UNCLASSIFIED