FINAL

Considerations in whether to use Marquardt Nonlinear Least Squares versus Lambert algorithm or both for NMD Cue Track Initiation (TI) calculations

Thomas H. Kerr III, Ph.D., Raytheon Consultant
TeK Associates
9 Meriam Street
Suite 7-R
Lexington, MA 02420-5312

Prepared for Raytheon Electronic Systems
528 Boston Post Road
Sudbury, MA 01776
Raytheon Contract Number A9HM-577955-F-830

28 September 2000

FINAL

Table of Contents

1. Introduction and problem statement

The quantities $\hat{\mathbf{X}}(t_2)$ and $P(t_2)$, as state and associated covariance, respectively, are being sought below for the purposes indicated in Section 2. These two quantities may be sequentially obtained using the Lambert solution methodology via Step **a** through Step **c** in Section 3 (with an addendum instrumental to its numerical evaluation offered in Section 4) to, respectively, first determine the velocity estimate at detection, then via the methodology in Section 5 to obtain the velocity component covariance which accompanies it by using two Jacobian transformations, as indicated.

An alternative approach that can be invoked is to apply the Marquardt (also known as the Levenberg-Marquardt) methodology of Sections 6 and 7 to obtain both the velocity estimate and its accompanying covariance at once. The Marquardt technique incurs a larger computational burden but provides a greater benefit by tolerating realistic antenna biases and provides corrected range as output even though the associated fence detection, in the case of NMD cues, doesn't use LFM pulse pairs to infer the appropriate Doppler correction compensation.

Indications from Section 8 are that J_2 does not need to be accounted for in either the Lambert or Marquardt algorithm because the additional complexity incurred to do so is great yet the benefit is apparently rather small since the effect appears to be too small to warrant analysts being so exacting in this aspect. Both techniques are presented below for explicit calculation of the two quantities being sought: a refined velocity estimate and its accompanying covariance to be used within Track Initiation (TI) for the tracking filter. Presenting these two alternatives in juxtaposition facilitates comparisons of benefits reaped versus drawbacks incurred.

Although many people may be familiar with the Lambert algorithm and the Lambert Theorem that spawned it from the historical literature [1], the specifics have now changed from being grounded in the following six familiar historical ephemeris parameters (defined in [16, p. 36]):

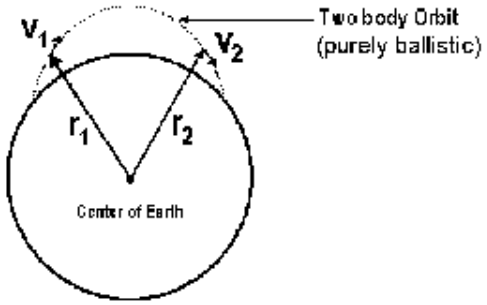
1. Semi-major axis,
2. Eccentricity,
3. Inclination,
4. Longitude of the ascending node,
5. Argument of periapsis,
6. Time of periapsis passage,

to just dealing with the velocities of direct relevance to the UEWR tracking as the bottom line. There is no longer a need to dwell on these orbital parameters per se or on universal variables [1], [3] since there are many other invariants in central force gravity fields that may be exploited and the relationship between instantaneous position and associated velocity of direct interest in UEWR TI suffices.

2. How results are to be used in the MAS SRS for track initiation with NMD cue

A Remote Sensor cue consists of an earlier ECI object state vector, $\hat{\mathbf{X}}(t_1)$, and some estimate of its observation error covariance, $C(t_1)$, after termination of powered (boost) flight (as passed to UEWR by BMC3 from either SBIRS or as a radar-to-radar hand-over).
The Lambert algorithm exploits a property of an inverse squared central force gravity field by allowing direct calculation of an accurate value of the target velocity at radar detection. The Lambert-based computed target velocity can be used as a so-called "warm start" or a "hot start" to initialize the UEWR tracking filter with a value that is closer to the true value (thus expediting accurate target tracking by taking less time to converge merely because it starts out being closer to its goal). The following procedure shall use the solution methodology referred to as "solving the Lambert problem" **(please see figure below)** for the trajectory characterization of a ballistic object, as obtained from two indicated position observations and the elapsed time-of-flight between those observations: t_2-t_1 (as the three inputs) to obtain the velocity $v^T(t_2)$ [as the output

Lambert Theorem states that for any two points r_1 and r_2 in inertial space and a time interval Δt (= t_2- t_1) between these points there exists a unique velocity at r_1, such that the orbit defined by r_1 and v_1 will pass through r_2 in time Δt.



enabling specification of $\hat{\mathbf{X}}(t_2)= [\mathbf{p}^T(t_2) \ \mathbf{v}^T(t_2)]^T$ to be used as the initial state to start the tracker] along with a much reduced accompanying covariance $P(t_2)$ as obtained from $C(t_2)$ and $C(t_1)$.
After being directed to the expected location in space and after obtaining a successful detection and verify, the R-, U-, and V-measurements of that same object at a more recent time t_2 allows an improved estimate of the object state and accompanying covariance to be computed according to the following sequence of steps:

i) Convert $R(t_2)$, $U(t_2)$, and $V(t_2)$ into an ECI position estimate, $\mathbf{p}(t_2)$, and an accompanying associated error covariance $C(t_2)$ corresponding to the detection time more recent than that of the original $C(t_1)$ associated with the cue [where these two covariances, so far, are just those associated with the position].

ii) From the value $\mathbf{p}^T(t_1)$ (extracted from the cue $\hat{\mathbf{X}}(t_1)= [\mathbf{p}^T(t_1) \ \mathbf{v}^T(t_1)]^T$), and from $\mathbf{p}(t_2)$ of previous step and knowledge of the time elapsed between the two sightings (t_2-t_1), use the iterative Lambert algorithm

(described below in Sections 3 and 4) related to the underlying ballistic trajectory to compute an estimate of the velocity $\mathbf{v}^T(t_2)$ (and a refined estimate $\mathbf{v}^T(t_1)$ for use in the subsequent endpoint Jacobian evaluations **of Section 5**).

iii) Compute the final tighter error covariance $P(t_2)$ for that object from the two constituents $C(t_1)$ and $C(t_2)$ (according to the Lambert procedure for covariances described below in **Section 5**).

iv) Transform the resultant pair, $\hat{\mathbf{X}}(t_2)$ and $P(t_2)$, to the ENU frame to be used for track initiation as a tracking filter initialization

The quantities $\hat{\mathbf{X}}(t_2)$ and $P(t_2)$ are to be obtained from the two separate Lambert related methodologies provided **next in Sections 3 to 5**. [Bold prose in this section denotes an addendum beyond what is in the current SRS.]

3. SRS specification of the Lambert algorithm for the velocity estimates

a. Specifying preliminary consolidating parameters before entering the actual algorithm:

$$r_1 \equiv |\mathbf{p_1}|;\ r_2 \equiv |\mathbf{p_2}|;\quad s \equiv \frac{1}{2}\left(p_1 + p_2 + |\mathbf{p_1} - \mathbf{p}_2|\right);\ \theta \equiv \arccos\left[\frac{\mathbf{p_1}\cdot\mathbf{p_2}}{|\mathbf{p_1}||\mathbf{p_2}|}\right]$$

$$x_o = \ell$$

Using the following further prescribed definitions for summarizing a collection of underlying pertinent parameters as being constant throughout the computation:

$$m \equiv \frac{8\mu(t_2 - t_1)^2}{\left(r_1 + r_2 + 2\sqrt{r_1 r_2}\cos\frac{1}{2}\theta\right)^3}$$

c. **Beginning of Lambert Solution Main Iteration Loop (i = 0, 1, 2, …, 9)**

$$\eta = \frac{x_i}{\left(\sqrt{1+x_i}+1\right)^2}$$

The number of significant digits of the answer relates to continued fraction levels used.

$$\xi(x_i) = \cfrac{8\left(\sqrt{1+x_i}+1\right)}{3+\cfrac{1}{5+\eta+\cfrac{(9/7)\eta}{1+\cfrac{(16/63)\eta}{1+\cfrac{(25/99)\eta}{1+\cfrac{(36/143)\eta}{1+\cfrac{(49/195)\eta}{1+\cfrac{(64/255)\eta}{1+\ldots}}}}}}}}$$

(The continued fraction expression is representation that is explained further in Section 4.)

$$h_1 = \frac{(\ell + x_i)^2(1 + 3x_i + \xi)}{(1 + 2x_i + \ell)[4x_i + \xi(3 + x_i)]};\quad h_2 = \frac{m(x_i - \ell + \xi)}{(1 + 2x_i + \ell)[4x_i + \xi(3 + x_i)]}$$

Now the fundamental two successive approximation iteration equations to be solved (as presented in [2]) are:

$$y_{i+1}^3 - y_{i+1}^2 - h_1\, y_{i+1}^2 - h_2 = 0$$

Since only the positive real root of the above equation for y_{i+1} is meaningful, attention is instead

$$x_{i+1} = \sqrt{\left(\frac{1-\ell}{2}\right)^2 + \frac{m}{y_i^2}} - \frac{1+\ell}{2} \quad \text{and}$$

redirected to the following alternative expression for the y_{i+1}, where there is a need to first evaluate

$$u = \frac{B}{2\left(\sqrt{1+B}+1\right)} \text{ where } B = \frac{27\,h_2}{4\left(1+h_1\right)^3}$$

$$y_{i+1} = \frac{1+h_1}{3}\left[2 + \frac{\sqrt{1+B}}{1+2\,u\,[K(u)]^2}\right]$$

where (from page 339 of [3]), the following continued fraction must be evaluated (where again the number of continued fraction levels retained relates directly to the number of significant digits available in the final answer):

$$K(u) = \cfrac{1/3}{1+\cfrac{(4/27)u}{1+\cfrac{(8/27)u}{1+\cfrac{(2/9)u}{1+\cfrac{(22/81)u}{1+\cfrac{(208/891)u}{1+\ldots}}}}}}$$

where the coefficients of the above continued fraction are defined to be

$$\gamma_{2n+1} = \frac{2\left(3n+2\right)\left(6n+1\right)}{9\left(4n+1\right)\left(4n+3\right)}$$

and

$$\gamma_{2n} = \frac{2\left(3n+1\right)\left(6n-1\right)}{9\left(4n-1\right)\left(4n+1\right)}$$

End of Lambert Solution Main Iteration Loop

The appropriate initial value of y used to start the above computations by looping between the two primary successive approximation iterations that together exhibit close to uniform convergence globally (in at most 9 iterations) without any singularities being present within this particular solution methodology is:

After completing the indicated 9 iterations of the loop specified above signifying convergence

$$y_o = \begin{cases} 0 \text{ if } \dfrac{h_2}{\left(1+h_1\right)^3} < -\dfrac{4}{27} \\ \dfrac{2}{3}\left(1+h_1\right) \text{if } \dfrac{h_2}{\left(1+h_1\right)^3} \geq -\dfrac{4}{27} \end{cases}$$

under all situations, the desired velocity at time t_2 (from page 307 of 1987 Battin) is:

$$v_2 = \sqrt{\frac{\mu}{a_m}}\left[x\left(i_2 \cdot i_{r_2}\right) i_2 + y\sqrt{\frac{r_2}{r_1}}\left(i_1 \cdot i_{r_1}\right) j_2 \right]$$

where

$$a_m = \frac{s}{2} = \frac{r_1 + r_2 + \left|\mathbf{r_1} - \mathbf{r_2}\right|}{4}$$

and the unit vectors j_1 and j_2 are in the directions of the minimum energy velocity vectors at the initial and terminal points. Let i_1 and i_2 be defined to make both i_1, j_1 and i_2 , j_2 be right handed and orthogonal coordinate systems.

Although a coarse velocity estimate is included in the original NMD Cue, a refined estimate of the initial velocity at time t_1 (from page 307 of [3]) may be calculated now as:

$$v_1 = \sqrt{\frac{\mu}{a_m}}\left[-x\left(i_1 \cdot i_{r_1}\right) i_1 + y\sqrt{\frac{r_2}{r_1}}\left(i_2 \cdot i_{r_2}\right) j_1 \right]$$

which is needed only for the Lambert-based evaluations of the accompanying covariance according to the procedure offered in the next section (and perhaps for improving the initialization of Marquardt, as discussed in Sections 6 and 7).

An alternative solution strategy is offered in [1, Chapter 5] (still in print). Although the alternative approach is intuitive and physically based, it is less specific at certain critical points, than the one presented here based on Richard Battin's (Charles Stark Draper Laboratory and MIT Aerospace Department) approach [2], [3]. For completeness, Stephen Nelson and Paul Zarchan (C. S. Draper Laboratory) have yet another more recent approach [4] to calculating the Lambert solution (also without accounting for presence of J_2) as all three of these approaches ignore. The approach of [4] is is also expressed directly in terms of position and velocity but, in general, requires a few more iterations than Battin's approach does. Battin's approach avoids historical singular situations that Gauss originally encountered in posing a solution and, moreover, Battin's algorithm appears to be uniformly convergent (i.e., the same number of iterations suffice to converge) no matter what starting condition is utilized (as long as it is consistent with what is recommended to be used for an initial guess for particular types of orbits or trajectories, where attention is confined here to just those being elliptical, as is the case for ballistic missiles and satellites).

4. Evaluating continued-fraction representation appearing in the Lambert algorithm

For a useful interpretation of continued fractions as hidden linear recurrence relations, please see Sec. 2.2.2 of [6]. The main idea is summarized below. A continued fraction of the following form (encountered twice above in Section 3):

$$f_n = a_o + \cfrac{b_1}{a_1 + \cfrac{b_2}{a_2 + \cdots + \cfrac{b_n}{a_n}}}$$

represents an equivalent recurrence in terms of two families (p and q for numerator and denominator) so that the general "nth convergent" above can be equivalently obtained from where p_n and q_n in the above final expression are obtained from the following two linear recurrence relations:

$$f_n = \frac{p_n}{q_n},$$

$p_n = a_n p_{n-1} + b_n p_{n-2}$ starting with $p_o = a_o$ and $p_1 = a_1 a_o + b_1$

and

$q_n = a_n q_{n-1} + b_n q_{n-1}$ starting with $q_o = 1$ and $q_1 = a_1$.

Rather than replace the two continued fraction expressions appearing in the main loop in Section 3 above and risk interfering with expected visual cross-checks by reviewers to the original as presented in Battin's work that wouldn't expect the equivalence spelled out yet, detailing the equivalence was postponed until now to be provided here. Please realize that these continued fractions are just short hand representations for additional secondary sub-loops within the Main Iteration Loop that must be evaluated on every iteration of the Main Loop index i. There are two different continued fractions above so there are, correspondingly, two secondary sub-loop evaluations called for inclusion in the Main Loop above.

5. SRS specification of the accompanying Lambert-based Covariance Calculation

The necessary derivatives to be used in determining the accompanying comparatively tight velocity covariance are numerically evaluated at the resulting solution to the Lambert problem (offered immediately above). This procedure requires prior evaluation of the two intermediate 3 x 3 transformation matrices that are to operate, respectively, on the two indicated position covariances on the right hand side below (denoted by the subscript p). This is done in order to obtain the desired tight final 3 x 3 velocity covariance result of primary interest here (denoted with subscript v), appearing on the left hand side below:

$$C_{\mathrm{V}}^{(\mathrm{det})} = T_C C_P^{(\mathrm{cue})} T_C^{\ T} + T_D C_P^{(\mathrm{det})} T_D^{\ T}$$

where

$C_P^{(\mathrm{cue})}$ = Cue ECI position covariance

$C_P^{(\mathrm{det})}$ = Fence (detection) ECI position covariance

where in the above, the indicated transformations to be used for pre- and post-multiplying the respective cue and detection covariances are the following two (3 x 3) Jacobian gradient matrices (with specific entries defined explicitly below for explicit evaluation in this role):

$$T_C = d(V_x^{(det)}, V_y^{(det)}, V_z^{(det)}) / d(P_x^{(cue)}, P_y^{(cue)}, P_z^{(cue)})$$

$$T_D = d(V_x^{(det)}, V_y^{(det)}, V_z^{(det)}) / d(P_x^{(det)}, P_y^{(det)}, P_z^{(det)})$$

Given a 6 x 1 state vector (P,V), the first and second derivatives of range with respect to time, rdot and racc, respectively, are computed as follows:

$$rdot = \frac{LOS \bullet V^{ER}}{rng}$$

$$racc = \frac{DV \bullet V^{ER} + LOS \bullet A^{ER}}{rng} - \frac{LOS \bullet V^{ER} \times LOS \bullet DV}{rng^3}$$

where

$$LOS = P - PS$$
$$PS = Position\_of\_Site\_in\_ECI$$
$$rng \equiv |LOS|$$
$$V^{ER} \equiv Earth\_\mathrm{Re}lative\_Velocity\_Vector$$
$$DV = V - VS$$
$$VS \equiv Velocity\_of\_Site\_in\_ECI$$
$$A^{ER} = A_G + \begin{matrix} + w * V_y \\ - w * V_x \\ 0 \end{matrix}$$
$$A_G = Gravitational\_Acceleration\_Vector$$
$$w = Earth\_rotation\_rate(radians / \sec)$$

The uncertainties in rdot and racc are then computed as follows:

$$\sigma rdot = sqrt(\left(\frac{\partial rdot}{\partial PV_i}\right) \times R \times \left(\frac{\partial rdot}{\partial PV_i}\right)^T)$$

$$\sigma racc = sqrt(\left(\frac{\partial racc}{\partial PV_i}\right) \times R \times \left(\frac{\partial racc}{\partial PV_i}\right)^T)$$

where $\left(\frac{\partial x}{\partial y_i}\right)$ represents the Jacobean of x with respect to the elements y_i and R is the ECI state vector covariance matrix.

$$\left(\frac{\partial rdot}{\partial PV_1}\right) = -rdot \times \frac{P_x - PS_x}{rng^2} + \frac{V_x + w \times P_y}{rng} - \frac{w \times (P_y - PS_y)}{rng}$$
$$\left(\frac{\partial rdot}{\partial PV_2}\right) = -rdot \times \frac{P_y - PS_y}{rng^2} + \frac{V_y - w \times P_x}{rng} - \frac{w \times (P_x - PS_x)}{rng}$$
$$\left(\frac{\partial rdot}{\partial PV_3}\right) = -rdot \times \frac{P_z - PS_z}{rng^2} + \frac{V_z}{rng}$$
$$\left(\frac{\partial rdot}{\partial PV_4}\right) = \frac{P_x - PS_x}{rng}$$
$$\left(\frac{\partial rdot}{\partial PV_5}\right) = \frac{P_y - PS_y}{rng}$$

$$\left(\frac{\partial rdot}{\partial PV_6}\right) = \frac{P_z - PS_z}{rng}$$

$$\left(\frac{\partial racc}{\partial PV_1}\right) = \frac{1}{rng} \times \left(-w \times (V_y - VS_y) + A_x^{ER} + (P_x - PS_x) \times \left(\frac{-U}{\|P\|^3} + \frac{U \times 3 \times P_x^2}{\|P\|^5}\right) + \right.$$

$$\left(P_y - PS_y\right) \times \frac{U \times 3 \times P_x \times P_y}{\|P\|^5} + \left(P_z - PS_z\right) \times \left.\frac{U \times 3 \times P_z \times P_x}{\|P\|^5}\right) +$$

$$\frac{(DV \bullet V^{ER} + LOS \bullet A^{ER}) \times (PS_x - P_x)}{rng^3} -$$

$$\frac{1}{rng^3} \times \left[\left(\left(V_x + w \times P_y\right) - w \times \left(P_y - PS_y\right)\right) \times LOS \bullet DV + V^{ER} \bullet LOS \times (V_x - VS_x)\right] -$$

$$(V^{ER} \bullet LOS \times LOS \bullet DV) \times \left(\frac{-3 \times (P_x - PS_x)}{rng^5}\right)$$

$$\left(\frac{\partial racc}{\partial PV_3}\right) = \frac{1}{rng} \times \left(\begin{array}{l} A_y^{ER} + (P_x - PS_x) \times \left(\dfrac{U \times 3 \times P_x \times P_z}{\|P\|^5}\right) + \\ \left(P_y - PS_y\right) \times \left(\dfrac{U \times 3 \times P_y \times P_z}{\|P\|^5} - \dfrac{U}{\|P\|^3}\right) + \left(P_z - PS_z\right) \times \left(\dfrac{U \times 3 \times P_z \times P_z}{\|P\|^5} - \dfrac{U}{\|P\|^3}\right) \end{array}\right) +$$

$$\frac{(DV \bullet V^{ER} + LOS \bullet A^{ER}) \times (PS_z - P_z)}{rng^3} -$$

$$\frac{1}{rng^3} \times \left[V_z \times LOS \bullet DV + V^{ER} \bullet LOS \times (V_z - VS_z)\right] -$$

$$(V^{ER} \bullet LOS \times LOS \bullet DV) \times \left(\frac{-3 \times (P_z - PS_z)}{rng^5}\right)$$

$$\left(\frac{\partial racc}{\partial PV_4}\right) = \frac{V_x + w \times P_y + V_x - VS_x - w \times (P_y - PS_y)}{rng} - \frac{\left(P_x - PS_x\right) \times \left(LOS \bullet DV + V^{ER} \bullet LOS\right)}{rng^3}$$

$$\left(\frac{\partial racc}{\partial PV_5}\right) = \frac{V_y - w \times P_x + V_y - VS_y + w \times (P_x - PS_x)}{rng} - \frac{\left(P_y - PS_y\right) \times \left(LOS \bullet DV + V^{ER} \bullet LOS\right)}{rng^3}$$

$$\left(\frac{\partial racc}{\partial PV_6}\right) = \frac{V_z + V_z - VS_z}{rng} - \frac{\left(P_z - PS_z\right) \times \left(LOS \bullet DV + V^{ER} \bullet LOS\right)}{rng^3}$$

The above partial derivatives are analytically provided here since they must be explicitly evaluated numerically at the two time points indicated (at time t_1 [cue] and at time t_2 [det]) in order that the above two 3 x 3 T-transformations be completely specified. Brian O'Dea warns that the above acceleration terms are no longer needed for our usage for TI calculations.
The above expressions for the derivatives appeared in IRAS and TD/SAT documentation and Kenneth Berkowitz (Raytheon) had previously encountered them in conjunction with the topic of *motion compensation* so they may already appear in the MAS SRS within the context of the

motion compensation topic. George Bohannon (XonTech) had indicated that over the years of conscientious XonTech revisions for various updates to TD/SAT and associated User Manuals, he believes that XonTech has weeded out and corrected all of the possible typos. I didn't want to risk introducing any new typos here so this is a copy of XonTech's original equations that are provided above.

Excerpt from UEWR SSTB documentation:

George Bohannon (XonTech) more recently uses slightly different notation for essentially the same description offered at the start of this Section 5 but still repeated here now for the reader's convenience:

"I wrote a section describing the use of the Lambert calculation for computing the covariance matrix for XonTech's pulse integration algorithm in SSTB. (The algorithm is called MCPI.) … I think that the notation that I now use below for the derivatives is perhaps clearer than the earlier notation that you extracted from one of our viewgraphs [sic, XonTech provided them by my request].

The trajectory constraint is imposed using a Lambert trajectory calculation. The Lambert calculation produces a velocity vector at one point on a trajectory given the position at two points on the trajectory and the difference between the times at those two points. The computer code that is used was written by Shio Oikawa (XonTech), and described in [9]. The application in [9] is quite different from MCPI, however, in that track initiation is performed using a detection in a search fence, whereas the MCPI application is prior to detection. Thus, the accuracy of the position in the radar range window (search fence or otherwise) for MCPI is generally lower than in the TI application, hence the estimated velocity is less accurate.

The Lambert method is also used for generating the covariance of the velocity estimate. These calculations are performed in ECI (Earth Centered Inertial) coordinates. To describe the covariance calculation, it is worthwhile expressing the Lambert calculation symbolically, as follows:

$$\vec{V} = \vec{L}(P^{(1)}, P^{(2)}, \Delta t)$$

where

$$\vec{V} = (V_1, V_2, V_3) = \textit{ECI velocity vector}$$

$$\vec{P}^{(i)} = (P_1^{(i)}, P_2^{(i)}, P_3^{(i)}) = \textit{ECI position vectors, } i = 1,\ 2$$

$$\Delta t = \textit{Time between the two po}\,\text{int}\,s$$

The subscripts 1, 2, and 3 designate the Cartesian components (x, y, and z, respectively) of the vectors. Trajectory point 1 is taken to be at booster burnout or object deployment, whereas point 2 is at the time of the potential radar detection.

Two transformation matrices are defined as follows:

$$M_{j,k}^{(i)} = \frac{\partial L_j(\vec{P}^{(1)}, \vec{P}^{(2)}, \Delta t)}{\partial P_k^{(i)}}$$

These derivatives are computed by one-sided finite-difference approximations. The velocity covariance is then

$$C^{(V)} = M^{(1)} C_1^{(P)} M^{(1)^T} + M^{(2)} C_2^{(P)} M^{(2)^T}$$

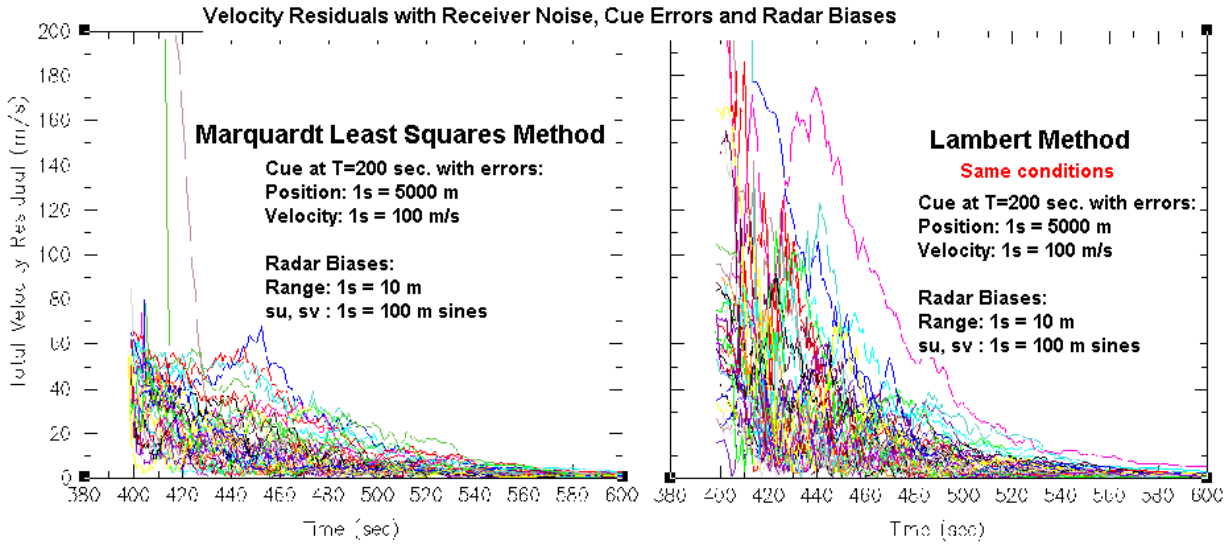
where $C_1^{(P)}$ and $C_2^{(P)}$ are the position covariance matrices at points 1 and 2 on the trajectory, and the superscript T indicates the matrix transpose."

Notice that all the terms included in the indicated Jacobians (now denoted by M instead of T) still need to be spelled out for implementation (as done here above) so that they may be

explicitly evaluated at the two end points. In real-time code, TeK Associates suspects that there will be less time available to do numerical perturbations (as one-sided difference approximations) to evaluate the necessary derivatives as XonTech has done in prior non-real-time simulations and still indicated here. That is what TeK Associates recommends changing above to instead just evaluate the stipulated derivatives (already spelled out analytically) at the two end points at cue and at detect.

6. Discussion of how Marquardt Nonlinear Least Squares may be used to obtain both refined velocity estimates and accompanying covariance simultaneously

Earlier demonstrations (see figure below depicting much smaller errors incurred by Marquardt than by Lambert for identical situation) of the benefits to NMD of using the Maquardt optimization technique included a demonstrated robustness to radar antenna biases (range: $1\sigma = 10$ m; σ_u, σ_v : $1\sigma = 100$ μ sines) that are otherwise unaccounted for but present nonetheless. More recent claims are that this Marquardt technique can also handle uncompensated Doppler measurements and automatically deduce corrected range without invoking LFM pulse pairs. This may be of interest since pulse pairs are not currently used with NMD cue detection; however, preliminary quantification demonstrates that ignoring range-Doppler coupling for Lambert problem is negligible since target range is from the earth center in these computations.



A drawback is that the well-known Marquardt optimization technique is generally a larger computational burden than otherwise encountered for Lambert-based solutions. (However, existing claims are that just using simple Lambert solution can't provide Doppler-corrected range without further iterations, a type of increased computational burden in itself for a variation on the standard Lambert solution approach that would be a greater computational burden than presented above for the straight forward single-shot Lambert approach). One variation that has been recommended is to first use the simple Lambert approach to obtain initial results then use these as inputs for the Marquardt approach as a more robust refinement thus incurring fewer iterations than otherwise encountered with Marquardt because the starting values are closer to the final solution. This proposed usage would mitigate the CPU burden of the Marquardt approach.

Regarding Brian O'Dea's claim in above concerning the benefits of using Marquardt in place of the Lambert algorithm and how its use ameliorates the deleterious effect of uncompensated range-Doppler coupling (so far, undocumented), Haywood Satz (Raytheon) reminded me that he and Duane Matthesien (Technia) looked into the order-of-magnitudes involved already. They had noticed that since the range-to-target used in the Lambert calculation is viewed from the center of the earth, the contribution due to the range-Doppler coupling effect with respect to the local radar is, perhaps, too small to care about. Please see quantification at the end of Section 8.

7. Specification of the Marquardt Algorithm (to eventually be pruned for the SRS) if use of Marquardt is in fact adopted by Raytheon

Inputs to the Marquardt algorithm are:

- Cue state vector of object and diagonal components of the accompanying cue covariance matrix;
- RAE measurements of object obtained in the radar fence, their associated three standard deviations, and time of detection so that the total time interval from cue to detection is provided.

Outputs of the Marquardt algorithm are:

- Refined state vector estimate and accompanying covariance matrix at detection time;
- The resulting 6x6 covariance matrix is, in general, fully populated (being devoid of pure zero elements).

The model to be fitted by the nonlinear least squares Marquardt algorithm represents the object in a ballistic trajectory and its path over time is of the following form:

$y = y(t; \mathbf{a})$ where $\mathbf{a}$ represents the identified unknown parameters to be fitted.

The unknown parameter **a** can be treated as a vector having M components. The fitting of the unknown parameters are computationally deduced in prescribed successive stages using the iterative Marquardt optimization procedure, as described in the five steps further below.
The scalar cost function to be minimized is of the following form to match an ideal ballistic template to the measured data by optimizing the parameters to obtain the best fit:

$$X^2(\mathbf{a}) = \sum_{i=1}^{2} \sum_{j=1}^{N_i} \left[\frac{y_{i,j} - y_j(t_i; \mathbf{a})}{\sigma_{i,j}} \right]^2$$

where N_1=6 and N_2=3 and $y(t_i; \mathbf{a})$ is a function that, for i=1, returns the parameters '**a**' (being the state vector being estimated) propagated to the time t_1 (the validity time of the state vector provided by the cue). For i=2, this function returns the parameters '**a**' propagated to the time t_2 (the radar measured detection time) [which is converted to the measurement coordinate system (RAE) where the range R includes the effects of range-Doppler coupling.
The UEWR-specific modifications to the otherwise standard Marquardt method detailed below are concisely summarized here now first:

- Dimensioning of y is 6 x 2 (actually its 6 x N) but here N is always 2.
- Providing a special routine "funcs" which returns the appropriate form of y [and it's associated] sensitivity] "dyda" depending upon whether "funcs" is called at the object cue time or at the radar fence object detection time.
- Modifying the routine MRQCOF to sum over j (as called for by the UEWR-revised cost function). As noted above, j runs to either N_1 (=6) or N_2 (=3).
- Since the scalar chi-square cost function includes differences of angles, resulting angle had to be compensated to ensure that if the |angle difference| was greater than 180 to either add or subtract 360, as appropriate to bring it within the correct span.

- Including an additional special array TFOB (denoting pulse length * frequency divided by bandwidth or "Tau F over B", as an obvious mnemonic notation) in the argument lists for existing MRQMIN and MRQCOF. This array is also passed to the routine "funcs".
- Modifying existing GAUSSJ0 to provide a return code that can be reflected back through the necessary argument lists and thus allow a more graceful robust response to possibly encountering a singular matrix.

The gradient of this scalar cost function with respect to the unknown parameters **a**, which should theoretically be zero at the "best fit" minimum, has the following components:

$$\frac{\partial \mathrm{X}^2}{\partial \mathrm{a}_\mathrm{k}} = -2\sum_{i=1}^{2}\sum_{j=1}^{\mathrm{N_i}} \frac{\left[y_{i,j} - y_j(t_i;\mathbf{a})\right]}{\sigma_{\mathbf{i,j}}^{\mathbf{2}}} \frac{\partial y_j(t_i;\mathbf{a})}{\partial \mathrm{a}_\mathrm{k}} \quad \text{for } k = 1, 2, \ldots, \mathrm{M}$$

Taking an additional partial derivative yields the following form for the Hessian:

$$\frac{\partial^2 \mathrm{X}^2}{\partial \mathrm{a}_\mathrm{k}\partial \mathrm{a}_\mathrm{l}} = 2\sum_{i=1}^{2}\sum_{j=1}^{N_i} \frac{1}{\sigma_{\mathrm{i,j}}^{2}} \left[\frac{\partial y_j(t_i;\mathbf{a})}{\partial \mathrm{a}_\mathrm{k}} \frac{\partial y_j(t_i;\mathbf{a})}{\partial \mathrm{a}_\mathrm{l}} - \left[y_{i,j} - y_j(t_i;\mathbf{a})\right] \frac{\partial^2 y_j(t_i;\mathbf{a})}{\partial \mathrm{a}_\mathrm{l}\partial \mathrm{a}_\mathrm{k}} \right]$$

Standard convention is to simplify the above and to eventually suppress any factors of 2 from appearing by utilizing the following two definitions for convenience:

$$\beta_k \equiv -\frac{1}{2}\frac{\partial^2 \mathrm{X}^2}{\partial \mathrm{a}_\mathrm{k}} \quad ; \quad \alpha_{kl} \equiv \frac{1}{2}\frac{\partial^2 \mathrm{X}^2}{\partial \mathrm{a}_\mathrm{k}\partial \mathrm{a}_\mathrm{l}}.$$

Using the above definitions along with the optimistic standard approximation that sufficiently close to the goal of achieving the minimum (and if the cost function is sufficiently smooth), then the cost function may be validly approximated by a quadratic form as:

$$\mathrm{X}^2(\mathbf{a}^*) \approx \gamma - \mathbf{d}^{\mathrm{T}}\mathbf{a}^* + \frac{1}{2}\mathbf{a}^{*T}\mathbf{D}\,\mathbf{a}^* \quad \text{(where constant } \gamma\text{, vector } \mathbf{d}\text{, and matrix } \mathbf{D} \text{ will be defined)},$$

and along with the fact that the cost function should be zero at this minimum, a quadratic form may be minimized by going from the current trial parameter $\mathbf{a}_{\mathrm{cur}}$ to the minimizing value $\mathbf{a}_{\mathrm{min}}$ in a single leap or step of the form:

$$\mathbf{a}_{\mathrm{min}} = \mathbf{a}_{cur} + \mathbf{D}^{-1} \cdot \left[-\Delta^2 \mathrm{X}^2(\mathbf{a}_{cur})\right]$$

Now re-expressing this single step equation, using the definition of the alpha's provided above, and defining the *curvature matrix* alpha as being one half of the Hessian, as

$$\boldsymbol{\alpha} = \frac{1}{2}\mathbf{D},$$

yields the following equivalent formulation as a set of linear equations that are to be solved for the increments $\delta\mathbf{a}$, which, when added to the current approximation of the parameter $\mathbf{a}_{\mathrm{cur}}$, provides the next approximation for $\mathbf{a}_{\mathrm{next}}$ as:

$$\sum_{l=1}^{M} \alpha_{kl}\,\delta \mathrm{a}_\mathrm{l} = \beta_k \quad \text{for } k = 1, 2, \ldots, \mathrm{M}$$

the result is

$$\mathbf{a} \leftarrow \mathbf{a} + \delta\mathbf{a}.$$

However, without assuming a quadratic approximation near the optimum because of possible pathologies in the surface of the scalar cost function, a more conservative pure steepest descent technique would require that the next step be down the gradient of the path of the following form:

$$\mathbf{a}_{next} = \mathbf{a}_{cur} - (\text{constant}) \cdot \left[\Delta \mathrm{X}^2(\mathbf{a}_{cur})\right],$$

which is equivalent, in this simplified notation, to

$\delta a_l = \text{constant} \times \beta_1$,

and the Marquardt algorithm is just a computational method used to force balance by embracing both of the above two "optimistic" and "conservative" approaches to offset each other, respectively, to gain an advantage by speeding up the attaining of the solution without incurring problems from a perhaps unwarranted (and unsubstantiated) assumption [also inconvenient to substantiate] *that near the optimum,* ***for this particular cost function being utilized and this path under investigation and this underlying physical structure assumed of inverse square gravity*** (WGS-84)*, the cost function is essentially a quadratic form* in this vicinity of the minimum. (That is, the cost function may have ridges or piecewise continuous jumps.)
This joint Marquardt objective is accomplished in two ways by first using this

$$\delta a_1 = \frac{1}{\lambda a_{ll}} \beta_l \quad \text{instead of the above } \partial a_1 = \text{constant} \times \beta_l,$$

and by further defining auxiliary primed variable alpha's as

$$\alpha'_{nn} \equiv \alpha_{nn}(1+\lambda); \text{ and } \alpha'_{nk} \equiv \alpha_{nk} \text{ for } n \neq k,$$

which may both be summarized by solving for the required δa_l from the following replacement system of linear equations:

$$\sum_{l=1}^{M} \alpha'_{kl}\, \delta a_1 = \beta_k .$$

Notice that for the auxiliary variable lambda much larger than 1, the optimization step size is cut down and the primed alpha's are diagonally dominant corresponding to an optimistic quadratic form methodology being invoked; while for lambda approaching zero, the pessimistic steepest descent methodology is invoked. This is the practical utility offered by the Marquardt algorithm as the "best of both worlds".
Given an initial guess for the set of fitted parameters **a**, the Marquardt procedure proceeds according to the following five Steps:

1. Compute the chi-squared variable (somewhat reminiscent of one from statistics):

   $X^2(\mathbf{a})$.

2. Initially, use a small value for the step-size scaling parameter as, say, lambda = 0.001.
3. Solve the following naturally arising linear equations:

   $$\sum_{i=1}^{M} \alpha'_{ki}\, \delta a_i = \beta_k \text{ for } \delta\mathbf{a} \text{ and evaluate } X^2(\mathbf{a}+\delta\mathbf{a}).$$

4. Proceed as follows:

   If $X^2(\mathbf{a}+\delta\mathbf{a}) \geq X^2(\mathbf{a})$, then increase lambda by a factor of 10, and then return to Step 3.

5. Proceed as follows:

   If $X^2(\mathbf{a}+\delta\mathbf{a}) < X^2(\mathbf{a})$, then decrease lambda by a factor of 10, update the trial solution $\mathbf{a} \leftarrow \mathbf{a}+\delta\mathbf{a}$, and return to Step 3.

The Marquardt algorithm is described in Section 15.5 of [7]. The corresponding Marquardt software code is in [8].

8. Evidence that use of J_2 to account for Oblateness of earth in gravity model can be ignored by both algorithms for TI

Consideration of J_2 is apparently unnecessary overkill in both Lambert and Marquardt for the UEWR application. Shio Oikawa (XonTech) comes to a similar conclusion **(see figure below)** in stating that his curves are almost the same whether or not he accounts for the presence of J_2.

Shio Oikawa's description of what he did to obtain the reported results (in figure below):

1. Propagated the Design-to trajectories to radar horizon break under J_2 gravity field (assumed to be the real world trajectory);
2. Noised up the initial BBO state (Booster Burn Out) to simulate the SBIRS measured position vector;
3. Noised up the radar horizon break time target (RV) to simulate the radar measurement: (Time between BBO and RV radar horizon break time: flight time, was kept constant = FLT);
4. Used the regular Lambert solution (Appendix I) to obtain BBO velocity which will get to the point obtained from Step #3;
5. Propagated the state from Step #4 for FLT;
6. Used the iteration method described below (FORTRAN code) until the error was less than 1m (if I remember correctly) at the radar horizon break All of the cases converged in 2 ~ 5 iterations which is very fast;
7. Compared (at radar horizon break time) the true velocity from procedure 1 to velocity obtained from Step #6 and kept the statistics;
8. Plotted the cumulative results.

Please note that plot shows the results of fitting tank BBO and RV radar horizon break state vectors (not RV deployment state to RV radar horizon break) using Lambert solution as an initial velocity vector and iterating until position error at the radar horizon break was less than a specified value.

Shio Oikawa's FORTRAN code for Step #4 above:

```
FLT  = TINT - TSTRT        ! flight time (Tstart, Tintercept)
                                        100  CONTINUE
      C
      C    Compute the velocity required at current position
      C
C         X: Current position, XT: intercept position
C         VRS: Lambert velocity at X
C
CALL LAMBRT(GM,TOL,X,VRS,XT,VT,FLT,R1MAG,V1MAG,R2MAG,V2MAG
     *,                    ANGD,F,G,FDOT,GDOT,I0,MAXITER,ITER,IERR)

IF (IERR .NE. 0) THEN
      WRITE(8,50)
      WRITE(8,*)' No Lambert solution:  Run terminated'
      END IF
                                                    C
C     Propagate the state to TINT under J2 gravity model
C
DO I = 1, 3
XJ2(I) = X(I)

VJ2(I) = VRS(I)

VR(I)  = VRS(I)

END DO
200    T    = 0.0
```

IEND = 0

ITER = ITER + 1

```
 220    IF ((T + DT) .GE. FLT) THEN
          IEND = 1              ! One more integration step
       DT  = FLT - T
       END IF
       T2   = T + DT
       CALL RK(XJ2,VR,XB,GM,REQ,J2,DTV,DT,A,TA,N,I0,I0) !Prop under J2
       IF (IEND .EQ. 1) GO TO 240
       T = T2
       GO TO 220
                                                          C
C      Compare the final target position
C
 240    DO I = 1, 3
          DIFP(I) = XT(I) - XJ2(I)  ! XT: true , XJ2 computed
        END DO
        DIFX = SQRT(DIFP(1)**2 + DIFP(2)**2 + DIFP(3)**2)

        IF (DIFX .GT. TOLX) THEN   ! TOLX = 1 meter
```

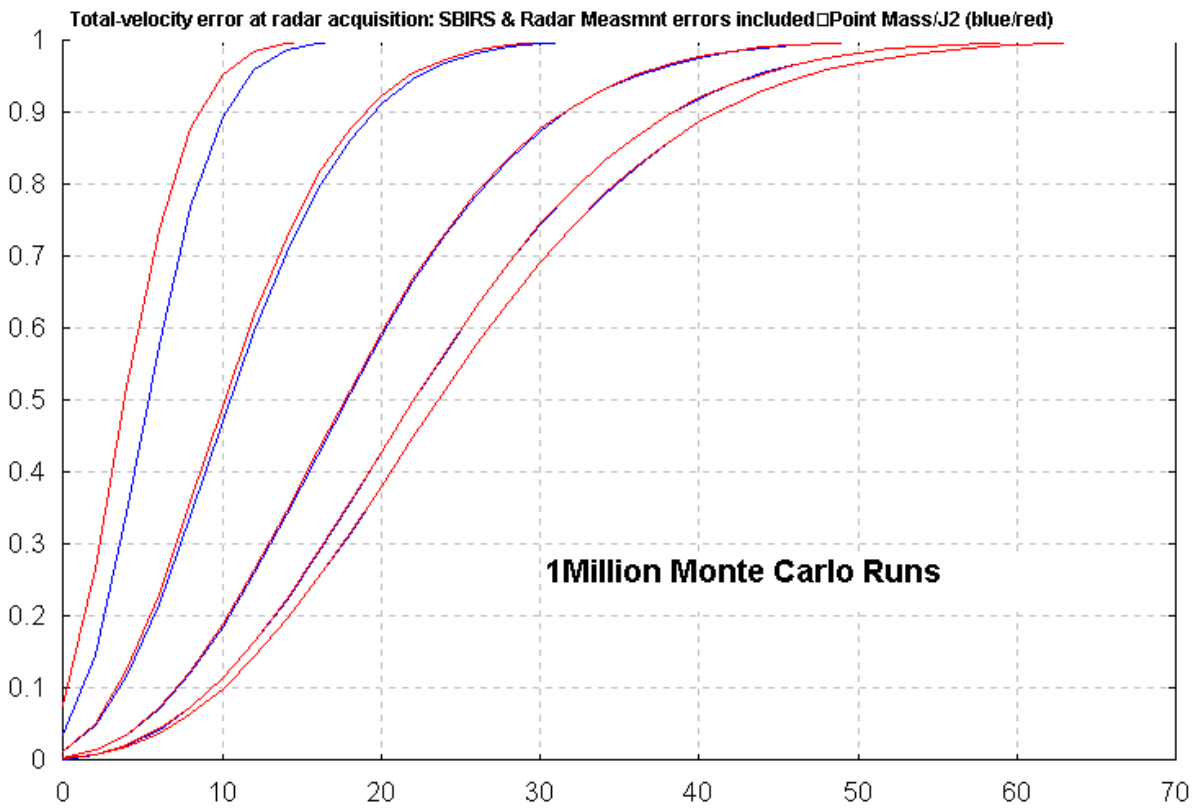
DO I = 1, 3

XJ2(I) = X(I)

```
       VR(I)  = VJ2(I) + DIFP(I) / FLT ! Adjust initial velocity
       VJ2(I) = VR(I)
```

END DO

```
       GO TO 200   ! Propagated position error > than tolerance
       END IF
       WRITE(9,30)FLT,DIFX,TOLX,ITER
```



Continuation of our observations regarding consequences of presence or absence of J_2

George Bohannon (XonTech) offers his perspective here:

"I wrote a section describing the use of the Lambert calculation for computing the covariance matrix for our pulse integration algorithm in SSTB. (The algorithm is called MCPI.) Perhaps it is of some help.

My experience has been that the Lambert method, as coded by Shio Oikawa, is very fast compared to Marquardt. This observation comes from my analysis of the estimation performance of the two methods for the MCPI application. However, since my goal [at the time did not include explicitly] comparing execution times, I don't have timing numbers.

The Lambert solution as implemented by Shio can be extended to include non-spherical earth and gravity models. Shio's basic Lambert code assumes a point-source gravity model and spherical earth. This assumption may or may not be adequate for the hot-start application, depending on propagation time and bandwidth. However, Shio has also implemented an iterative technique for refining the solution to include the J_2 contribution, which I suspect could also be used to include still higher order terms. Shio indicated that it typically converges to less than 1 meter of position error in 5 iterations or fewer. There is also an iterative extension to the SSTB implementation to include non-spherical terms, but I have found it to be extremely slow. I've seen code for both extensions, and they are different, but I cannot say how they compare in terms of execution time; I haven't actually tried using Shio's extension. Of course, you also know about the Vinti method for including J_2."

Resuming TeK Associates' view of the issues associated with presence or absence of J_2:

Tek Associates is aware of the (late) John Vinti's book [5] and viewed its associated 1993 TRW Fortran software that does account for J_2 but at earlier joint meeting with XonTech and Raytheon

in the summer of 1999, Fredrick Daum (Raytheon) revealed that such a detailed treatment would only change the final computed result by about 5 m/sec. Thus, this apparently changes the final answer by too small amount to really worry about for UEWR in Fred Daum's opinion (which TeK Associates has also adopted). Shio Oikawa's more recent simulations presented above have only confirmed this view.

The much greater computational complexity incurred without significant benefits being apparent to UEWR drove me to embrace the simpler implementation (without J_2 present). Vinti's approach [5] is completely different from Battin's [2], [3] or Bate and Mueller [1] and more complex.

None of XonTech's earlier PowerPoint presentations (by Shio Oikawa on 7-28-99 [9], by Teresa Caampued on10-15-99 [10], by Brian O'Dea and George Bohannon on 9-30-99 and 10-1-99 [11]) utilized J_2 for the Lambert or Marquardt computations even though they acknowledged familiarity with the existence of the Vinti results [5].

Neither the Fortran computer code (based on 1971 Bate and Mueller [1]) that Shio Oikawa sent (Appendix I) nor the version of the Lambert algorithm that is written up here in Section 3 for the SRS accounted for earth's oblateness (by including J_2) following the 1984 and 1987 prescription for it [2], [3] by Richard Battin include a consideration of J_2. I hope that this current absence of J_2 considerations in the Lambert algorithm for the SRS won't be a problem for Raytheon.

When and where J_2 is apparently important is in the Lambert calculations for a planetary rendezvous, where the mission time is fairly long and these secondary considerations mount up (over time) to have a significant impact that can't be ignored. For this, the use of [5] is necessary.

There are other situations where inclusion of J_2 is important such as in the tracking filter and in the BLS "batch" filter. Fred Daum (as well as all of the legacy SRS's) have always advocated its use there. Raytheon historical literature quotes that in performing tracking, absence of the J_2 term in the associated tracking model used by the tracking filter can lead to as much as 10 nm error in estimated impact point (after 30 minutes). We have been including J_2 where we think it is really necessary for UEWR.

Brian O'Dea's original (9/8/00) description of Xontech's particular customization, which TeK Associates used to provide the discussion of Marquardt in Sections 6 and 7:

The chisq merit function is:

$$\chi^2(a) = \sum_{i=1}^{2} \sum_{j=1}^{n_i} \left[\frac{y_{i,j} - y_j(t_i; a)}{\sigma_{i,j}} \right]$$

where n_1=6 and n_2=3.

$y(t_I;a)$ is a function that for i=1 returns the parameters 'a' (the SV being estimated) propagated to the time t_1 (the validity time of the cue provided SV). For i=2 this function returns the parameters 'a' propagated to the time t_2 (the measurement time) and converted to the measurement coordinate system (RAE in my case) where R includes the effects of range-Doppler coupling.

Thus the modifications to the Marquardt method provided by numerical recipes were to:

- Dimension 'y' 6 x 2 (actually 6 x n) but in practice 'n' is always 2.
- Provide a routine 'funcs' which returns the appropriate form of 'y' and 'dyda' depending upon whether funcs is called at the cue time or the measurement time.

- Modify the routine MRQCOF to sum over j (as called for by the merit function). As noted above 'j' runs to either 6 or 3.
- As the chisq merit function includes differences of angles I had to ensure that if the |angle difference| was greater than 180 to add or subtract 360 as appropriate.
- Include an array TFOB (implying pulse length * frequency divided by bandwidth or 'tau F over B') in the argument lists for MRQMIN and MRQCOF. This array is passed to the routine 'funcs'.

I also modified GAUSSJ0 to provide a return code that could be carried back thru the necessary argument lists and thus allow a more graceful response to encountering a singular matrix.
The only missing element is how to compute dyda at the two times. In the October [1999] presentation I provided the details about how I compute dyda at the cue provided SV validity time. Of course, my dyda was derived based on my propagation method (ECI) whereas I believe that the UEWR propagator will operate in ENU coordinates.
TeK Associates recent (9/23/00) request to Brian O'Dea for more elaboration on specifics:
Section 9 now more fairly depicts the situation…. I no longer believe that I will need more detail on the specifics of what is in the subroutines that you referred me to until a decision is made by the Raytheon/XonTech team to actually include Marquardt in the SRS. The ball is in their court now….
I do still need a little more explanation of how covariance is obtained or extracted from Marquardt and why cost function was customized in the manner indicated as a slight departure from the standard.
Can you (please will you) shed a little more light on how you incorporate the gravity model into your use of Marquardt?
Also, you seem to have said two different parameter "a's" are possible outputs, depending on whether we want at cue or at detection. Please elaborate (or clarify) how that fits in (or departs from) the standard Marquardt methodology.
Brian O'Dea's more detailed views of this entire discussion (received on 9/26/00):
Comments in no particular order.

1. Regarding section 5: the partial derivatives that are analytically presented in that section have nothing to do with the transformation matrices Tc and Td. They should be removed from that section. [In each of your earlier drafts I was sure that your intent was to remove those partials and thus did not pipe up as strongly or as early as I should have.] Note however that the first of those two partial derivatives (partial of rdot function with respect to an ECI SV) is used by the Marquardt technique. The latter of those two partial derivatives (partial of racc function with respect to an ECI SV) is not used by the Marquardt technique but is used in Motion Compensation.
2. You ask for an explanation of how covariance is extracted from Marquardt. Unfortunately I can not embellish upon the description provided in Numerical Recipes. NR defines the covariance as the inverse of the alpha matrix and then provides a subsequent section (14.5 in the 1990 FORTRAN Version[15]) that provides more details about calculating confidence intervals. [As an aside, I used the techniques in 14.5 to derive confidence intervals for use in Motion Compensation.]
3. The cost function was customized as indicated in order to account for the multidimensional nature of the measurements (px, py, pz, vx, vy, vz and r, a, e). If you recall, the equations that NR provides that are related to 'y' (e.g., 14.4.5) consist of only a single dimension. In hindsight (i.e., something I never did) I believe I could have further modified the chisq merit function to utilize full covariance matrices rather than just the diagonal elements.
4. I reference the gravity model during propagation of the SV being estimated (i.e., the parameters of interest, 'a') to the times of the "measurements" (i.e., to the cue time and the detection time). Note that I use an Extended Trapezoidal Corrector for propagation. For this purpose (propagation) I use a gravity model than includes 8 spherical and 8 tesseral terms. Actually the # of terms is specified by the user but I believe that in practice we run with 8,8. I also reference the gravity model for the purpose of computing the derivatives of the propagation function with respect to the parameters of interest. However, for the purpose of computing the aforementioned

derivatives, I make some simplifications: (1) I calculate the derivatives as though the propagation is performed using the simple Extended Euler technique. (2) I calculate the derivatives as though the Earth is a point source (no asymmetries). The presentation that George and I gave on 9/30 and 10/1 includes slides (Tracker, pages 12-16 now offered below) that provide more details about the calculation of the derivatives.

5. One can define the propagation and the calculation of the partial derivatives in such a manner as to place the parameters and associated covariance at any time of interest. When I first started working with the Marquardt method I defined the propagation and derivatives so that the parameters were estimated at the time of the first measurement. In general I'd then have to propagate those parameters (the estimated SV) and the accompanying covariance matrix to the time of the last measurement before use. For the Marquardt method that we implemented in the PAVE PAWS Mission Software (non bonded), I defined the propagation and derivatives so as to provide a SV and covariance valid at the time of the last measurement. Finally, note that I have compared the SV and covariance that one gets by estimating them at the time of the first measurement and then propagating them to the time of the last measurement with the SV and covariance that one gets by estimating them at the time of the last measurement and I've found them to be identical.

Marquardt Method Review
Derivation of $\frac{\partial ECI(t_i, sv)}{\partial sv_k}$ (1 of 3)

- Definitions

$$ECI(t, \overrightarrow{SV}) = \left\{P\left(t, \overrightarrow{SV}\right), V\left(t, \overrightarrow{SV}\right)\right\}$$

$$P(t, \overrightarrow{SV}) \approx \vec{P}_n = \vec{P}_{n-1} + \Delta t \vec{V}_{n-1} + \frac{\Delta t^2}{2} \mu \frac{\vec{P}_{n-1}}{\left\|\vec{P}_{n-1}\right\|^3}$$

$$V\left(t, \overrightarrow{SV}\right) \approx \vec{V}_n = \vec{V}_{n-1} + \Delta t \mu \frac{\vec{P}_{n-1}}{\left\|\vec{P}_{n-1}\right\|^3}$$

$$\left\|\vec{P}_0\right\|^3 = \left(P_{x,0}^2 + P_{y,0}^2 + P_{z,0}^2\right)^{3/2}$$

$$\Delta t \equiv \frac{t}{n}$$

$$\vec{P}_n \equiv \left(P_{x,n}, P_{y,n}, P_{z,n}\right)$$

- Therefore

$$\frac{\partial ECI\left(t_i, \overrightarrow{SV}\right)}{\partial SV_k} = \begin{pmatrix} \frac{\partial P_{x,n}}{\partial P_{x,0}} & \frac{\partial P_{x,n}}{\partial P_{y,0}} & \cdots & \frac{\partial P_{x,n}}{\partial V_{z,0}} \\ \frac{\partial P_{y,n}}{\partial P_{x,0}} & \cdots & & \frac{\partial P_{y,n}}{\partial V_{z,0}} \\ \vdots & & & \\ \frac{\partial V_{z,n}}{\partial P_{x,0}} & \cdots & & \frac{\partial V_{z,n}}{\partial V_{z,0}} \end{pmatrix}$$

Marquardt Method Review
Derivation of $\frac{\partial ECI(t_i, sv)}{\partial sv_k}$ (2 of 3)

- Derivatives are Computed Iteratively

$$\frac{\partial P_{i,n}}{\partial P_{j,0}} = \frac{\partial}{\partial P_{j,0}}\left(P_{i,n-1}\right) + \Delta t \frac{\partial}{\partial P_{j,0}}\left(V_{i,n-1}\right) + \frac{\Delta t^2 \mu}{2} \frac{\partial}{\partial P_{j,0}}\left(\frac{P_{i,n-1}}{\left\|\vec{P}_{n-1}\right\|^3}\right) \qquad \frac{\partial V_{i,n}}{\partial V_{j,0}} = \frac{\partial V_{i,n-1}}{\partial V_{j,0}} + \Delta t \mu \frac{\partial}{\partial V_{j,0}} \frac{\vec{P}_{n-1}}{\left\|\vec{P}_{n-1}\right\|^3}$$

$$\frac{\partial V_{i,n}}{\partial P_{j,0}} = \frac{\partial}{\partial P_{j,0}}\left(V_{i,n-1}\right) + \Delta t \mu \frac{\partial}{\partial P_{j,0}} \frac{\vec{P}_{n-1}}{\left\|\vec{P}_{n-1}\right\|^3} \qquad \frac{\partial}{\partial P_{j,0}}\left(\frac{P_{i,n-1}}{\left\|\vec{P}_{n-1}\right\|^3}\right) = \frac{\partial P_{i,n-1}/\partial P_{j,0}}{\left\|\vec{P}_{n-1}\right\|^3} - \frac{3P_{i,n-1}\left(\vec{P}_{n-1} \bullet \frac{\partial \vec{P}_{n-1}}{\partial P_{j,0}}\right)}{\left\|\vec{P}_{n-1}\right\|^5}$$

$$\frac{\partial P_{i,n}}{\partial V_{j,0}} = \frac{\partial}{\partial V_{j,0}}\left(P_{i,n-1}\right) + \Delta t \frac{\partial}{\partial V_{j,0}}\left(V_{i,n-1}\right) + \frac{\Delta t^2 \mu}{2} \frac{\partial}{\partial V_{j,0}}\left(\frac{P_{i,n-1}}{\left\|\vec{P}_{n-1}\right\|^3}\right) \qquad \frac{\partial}{\partial V_{j,0}}\left(\frac{P_{i,n-1}}{\left\|\vec{P}_{n-1}\right\|^3}\right) = \frac{\partial P_{i,n-1}/\partial V_{j,0}}{\left\|\vec{P}_{n-1}\right\|^3} - \frac{3P_{i,n-1}\left(\vec{P}_{n-1} \bullet \frac{\partial \vec{P}_{n-1}}{\partial V_{j,0}}\right)}{\left\|\vec{P}_{n-1}\right\|^5}$$

$$i, j \in [x, y, z]$$

Marquardt Method Review
Derivation of $\frac{\partial ECI(t_i, sv)}{\partial sv_k}$ (3 of 3)

- Starting with the First Iteration

$$\begin{pmatrix} \partial \boldsymbol{P}_{x,+} & \partial \boldsymbol{P}_{x,+} & \cdots & \partial \boldsymbol{P}_{x,+} \\ \partial \boldsymbol{P}_{x,0} & \partial \boldsymbol{P}_{x,0} & & \partial \boldsymbol{V}_{x,0} \\ \partial \boldsymbol{P}_{x,+} & \cdots & & \partial \boldsymbol{P}_{x,+} \\ \partial \boldsymbol{P}_{x,0} & & & \partial \boldsymbol{V}_{x,0} \\ \vdots & & & \\ \partial \boldsymbol{V}_{x,+} & \cdots & & \partial \boldsymbol{V}_{x,+} \\ \partial \boldsymbol{P}_{x,0} & & & \partial \boldsymbol{V}_{x,0} \end{pmatrix} = \begin{pmatrix} 1 + \frac{\Delta t^2 \mu}{2}\left(\frac{1}{\left\|\vec{\boldsymbol{P}}\right\|^3} - \frac{3\boldsymbol{P}_{x,+}^2}{\left\|\vec{\boldsymbol{P}}\right\|^5}\right) & \frac{\Delta t^2 \mu}{2}\left(\frac{-3\boldsymbol{P}_{x,+}\boldsymbol{P}_{y,+}}{\left\|\vec{\boldsymbol{P}}\right\|^5}\right) & \frac{\Delta t^2 \mu}{2}\left(\frac{-3\boldsymbol{P}_{x,+}\boldsymbol{P}_{z,+}}{\left\|\vec{\boldsymbol{P}}\right\|^5}\right) & \Delta t & 0 & 0 \\ \frac{\Delta t^2 \mu}{2}\left(\frac{-3\boldsymbol{P}_{x,+}\boldsymbol{P}_{y,+}}{\left\|\vec{\boldsymbol{P}}\right\|^5}\right) & 1 + \frac{\Delta t^2 \mu}{2}\left(\frac{1}{\left\|\vec{\boldsymbol{P}}\right\|^3} - \frac{3\boldsymbol{P}_{y,+}^2}{\left\|\vec{\boldsymbol{P}}\right\|^5}\right) & \frac{\Delta t^2 \mu}{2}\left(\frac{-3\boldsymbol{P}_{y,+}\boldsymbol{P}_{z,+}}{\left\|\vec{\boldsymbol{P}}\right\|^5}\right) & 0 & \Delta t & 0 \\ \frac{\Delta t^2 \mu}{2}\left(\frac{-3\boldsymbol{P}_{x,+}\boldsymbol{P}_{z,+}}{\left\|\vec{\boldsymbol{P}}\right\|^5}\right) & \frac{\Delta t^2 \mu}{2}\left(\frac{-3\boldsymbol{P}_{y,+}\boldsymbol{P}_{z,+}}{\left\|\vec{\boldsymbol{P}}\right\|^5}\right) & 1 + \frac{\Delta t^2 \mu}{2}\left(\frac{1}{\left\|\vec{\boldsymbol{P}}\right\|^3} - \frac{3\boldsymbol{P}_{z,+}^2}{\left\|\vec{\boldsymbol{P}}\right\|^5}\right) & 0 & 0 & \Delta t \\ \Delta t \mu\left(\frac{1}{\left\|\vec{\boldsymbol{P}}\right\|^3} - \frac{3\boldsymbol{P}_{x,+}^2}{\left\|\vec{\boldsymbol{P}}\right\|^5}\right) & \Delta t \mu\left(\frac{-3\boldsymbol{P}_{x,+}\boldsymbol{P}_{y,+}}{\left\|\vec{\boldsymbol{P}}\right\|^5}\right) & \Delta t \mu\left(\frac{-3\boldsymbol{P}_{x,+}\boldsymbol{P}_{z,+}}{\left\|\vec{\boldsymbol{P}}\right\|^5}\right) & 1 & 0 & 0 \\ \Delta t \mu\left(\frac{-3\boldsymbol{P}_{x,+}\boldsymbol{P}_{y,+}}{\left\|\vec{\boldsymbol{P}}\right\|^5}\right) & \Delta t \mu\left(\frac{1}{\left\|\vec{\boldsymbol{P}}\right\|^3} - \frac{3\boldsymbol{P}_{y,+}^2}{\left\|\vec{\boldsymbol{P}}\right\|^5}\right) & \Delta t \mu\left(\frac{-3\boldsymbol{P}_{x,+}\boldsymbol{P}_{z,+}}{\left\|\vec{\boldsymbol{P}}\right\|^5}\right) & 0 & 1 & 0 \\ \Delta t \mu\left(\frac{-3\boldsymbol{P}_{x,+}\boldsymbol{P}_{z,+}}{\left\|\vec{\boldsymbol{P}}\right\|^5}\right) & \Delta t \mu\left(\frac{-3\boldsymbol{P}_{y,+}\boldsymbol{P}_{z,+}}{\left\|\vec{\boldsymbol{P}}\right\|^5}\right) & \Delta t \mu\left(\frac{1}{\left\|\vec{\boldsymbol{P}}\right\|^3} - \frac{3\boldsymbol{P}_{z,+}^2}{\left\|\vec{\boldsymbol{P}}\right\|^5}\right) & 0 & 0 & 1 \end{pmatrix}$$

TeK Associates further explanation of its prior view (9/26/00):

Thanks. This was what I was looking for. I'll append your response to my report (as I checked to see that it was acceptable to you). I could never paraphrase it as well as you conveyed it.

Part of my problem was in not actually having attended the 30 Sept-1 Oct. 1999 video conference XonTech presentation because there wasn't enough room here and I was asked to not go by Raytheon. Instead, I had to rely on the interpretation of these derivatives (some months later) from another local XonTech representative who was there but is usually less concerned about these specifics so unfortunately your material was filtered for me by the "wrong eyes" on these aspects. [Perhaps it was actually a benefit for me in disguise since I wasn't lulled into complacency by pictures but still sought a written description rather than just a high level summary in PowerPoint or Adobe Acrobat slides.]

TeK Associates didn't know whether the acceleration partial was part of an unstated or implied "chain-rule" with other critical terms not yet being fully elaborated. The inverse square acceleration of gravity is present even during a potential target's ballistic regime. Without further elaboration, we wouldn't know the intent of the presence of these two types of terms: differentiation with respect to "rdot" and with respect to "racc". As with most things, what is obvious to the person doing it is not obvious to others until it is explicitly stated or written to accompany it.

We have numbers for the effect of Doppler Coupling compensation using standard 300 KHz detection waveform, a nominal carrier center frequency of 435 MHz for the LFM pulse, and an anticipated velocity error of 1 km/sec yielding 23 km range errors. It is stated on page 278 of [13] that for PAVE PAWS, the range-Doppler Coupling (RDC) coefficient is 36 secs. According to Fred Daum, UEWR RDC is typically 1 sec. (please see page 529 of [14]) so worse case range error would be 5 km for a 5 km/sec target and 7 km for a 7 km/sec target. After obtaining this error with respect to the local radar antenna, we claim that the result is very small as compared to the target range with respect to the center of the earth (that is used in the actual Lambert computations by adding the radius of the earth ~ 6378.1 km [equatorial]). It is even less of a worry if a proposed Doppler waveform is used. We also assume the tracking bandwidth to be 10 MHz (as Tuley is and as has been proposed as a common upgrade for all of UEWR).

9. Summary/Monday Morning Quarterbacking

The write-up of Lambert in Section 3 is more like the usual style of an SRS except for the presence of two continued fractions. The use of continued fractions is just short-hand representation for a pair of linear finite recursion equations.

These two continued fractions could have been replaced there with the equivalent two iteration equations at both locations in Section 3 where continued fractions occur (but this could possibly interfere with some system analyst cross-checking these results back to the original Battin 1984 paper [2] and 1987 textbook [3]) since Battin only uses the continued fraction representation there. Instead I gave how to evaluate continued fractions in Section 4 following the Lambert section. The two continued fractions are essentially two embedded sub-loops nestled within the single Main Loop of the Lambert algorithm.

My quandary was "should I be clearer for the manager/system analyst" (who wants to be able to easily cross-check it) or should I be clearer for the programmer (who must implement it). I decided, that temporarily, I would cater to the former. Then let them tell me to change it for the latter (after they had checked it).

The pages of derivatives appearing in my Section 5 on Lambert-based covariance evaluation are needed for calculating the Jacobian that is used to pre- and post-multiply other covariance matrices to obtain the final answer. These expressions for the derivatives appeared in IRAS and TD/SAT documentation and Ken Berkowitz (Raytheon) had encountered them in conjunction with motion compensation. George Bohannon (XonTech) had indicated that over the years of revision for various TD/SAT updates, he believes that they have

weeded out all of the typos. I didn't want to risk introducing any new typos of my own here so I used XonTech's original equations here.

In talking to Brian O'Dea (XonTech), he strongly favored use of Marquardt over Lambert

because it solved a host of other problems. Since Marquardt is a larger computational burden, I was worried about its use in real-time operational software. However, Lambert can be used to initialize Marquardt, and by starting closer to the eventual Marquardt solution thereby cut down on the number of iterations needed by Marquardt to converge to its solution.

To date, I figured out the Lambert stuff myself and wrote it up as Sections 3 and 4 and explained how the continued fraction representation is computationally evaluated and what it means. I have received exemplary recent support from XonTech about the specifics of a Marquardt implementation by their referring me to Section 15.5 of [7] and having later received more detail from them on how XonTech customizes the cost function they use, which departs slightly from the standard one depicted in [7].

I have presented the case for using either the Lambert algorithm or the Marquardt algorithm or both in the role of specifying TI for NMD cue. The Raytheon/Xontech team needs to decide what it prefers to pursue in this real-time role to be reflected in the SRS. If the team decides to use Marquardt, then more detail needs to be supplied on it for the SRS. Perhaps there is no need to get 'hot and bothered" about acknowledged deficiencies in the write-up of the Marquardt algorithm before we know for sure that it will be used in the SRS. Someone needs to decide whether UEWR MAS SRS will use Lambert algorithm, the Marquardt algorithm, or both together.

Appendix I: XonTech's Lambert algorithm FORTRAN code (based on [1])

According to Shio Oikawa, "In my analysis, I used the algorithm from R. Bate, D. Mueller, and J. White as Chapter 5 of [1] since it looked easier to implement at the time. As long as this (Lambert solution) uses the point mass (no J_2), any valid method should give the same results."

```
&INPUT
GM = 3986005.E8
    R1  = -2080537.353,-1331784.494,6039421.362,
    R2  = -1803306.13533542,-412620.944243097,6343961.79867607,
    DT = 138.8808
    TOL = 1.E-15
    MAXITER = 1000
    ISWT = 0
    TOL = 1.E-13
&END
C     ***************************************************************
SUBROUTINE LAMBRT(GM,TOL,R1,V1,R2,V2,DT,R1MAG,V1MAG,R2MAG,V2MAG
*,            ANGD,F,G,FDOT,GDOT,ISWT,MAXITER,ITER,IERR)
IMPLICIT REAL*8(A-H,O-Z)
DIMENSION R1(3),V1(3),R2(3),V2(3)
    PI = ACOS(-1.0)             ! Value of Pi (= 3.1415...)
    DR = 180.0 / PI             ! Rad. to Deg. conversion factor
IERR = 0
IF (ISWT .EQ. 0) THEN
                              R1MAG = SQRT(R1(1)**2 + R1(2)**2 + R1(3)**2)
                              R2MAG = SQRT(R2(1)**2 + R2(2)**2 + R2(3)**2)
                            DOTR  = R1(1) * R2(1) + R1(2) * R2(2) + R1(3) * R2(3)
     ANGR  = ACOS(DOTR/(R1MAG*R2MAG))    ! Angle between two
     ANGD  = ANGR * DR                   ! position vectors
```

```
ELSE
                                        ANGR  = ANGD / DR
END IF
    A    = SQRT(R1MAG * R2MAG * (1.0 + COS(ANGR)))
    Z    = 0.00001
    ITER = 0
                                          C
C     Compute the value of C and S using the summation.
                                          C
100  COLD  = 0.0
DO I  = 1, 100
                                        UP  = (-Z)**(I-1)
                                       IDN = 2 * (I-1) + 2
DN  =  FACT(IDN)
                                    CNEW = COLD + UP / DN
      IF (ABS(CNEW - COLD) .LE. TOL) GO TO 200
      COLD = CNEW
END DO
                                     200 C = CNEW
SOLD  = 0.0
DO I  = 1, 100
                                        UP  = (-Z)**(I-1)
                                       IDN = 2 * (I-1) + 3
DN  =  FACT(IDN)
                                    SNEW = SOLD + UP / DN
      IF (ABS(SNEW-SOLD) .LE. TOL) GO TO 210
      SOLD = SNEW
END DO
                                     210 S = SNEW
IF (C .LT. 0.0) GO TO 800
Y = R1MAG + R2MAG - A * (1.0 - Z * S) / SQRT©
IF (Y .LT. 0.0) GO TO 800
X = SQRT(Y / C)
F = X * X * X * S + A * SQRT(Y) - SQRT(GM) * DT
      CZOLD = 0.0
      DO I = 1, 100
                                        UP  = (-Z)**(I-1)
                                        IDN = 2 * I + 2
DN =  FACT(IDN)
                                  CZNEW = CZOLD - FLOAT(I) * UP / DN
          IF (ABS(CZNEW-CZOLD) .LE. TOL) GO TO 220
          CZOLD = CZNEW
END DO
                                     220 DCDZ = CZNEW
SZOLD = 0.0
DO I = 1, 100
                                        UP  = (-Z)**(I-1)
                                        IDN = 2 * I + 3
DN =  FACT(IDN)
                                  SZNEW = SZOLD - FLOAT(I) * UP / DN
      IF (ABS(SZNEW-SZOLD) .LE. TOL) GO TO 230
```

```
    SZOLD = SZNEW
END DO
                                        230 DSDZ = SZNEW
DYDZ = A * ((S + Z * DSDZ)/SQRT© + 0.5 * (1.0 - Z * S)
*     * DCDZ / SQRT(C**3))
DXDZ = 0.5 * (DYDZ/SQRT(C*Y) - SQRT(Y/C**3) * DCDZ)
DFDZ = 3.0 * X * X * DXDZ * S + X * X * X * DSDZ
*     + 0.5 * A * DYDZ / SQRT(Y)
ZNEW = Z - F / DFDZ
    ITER = ITER + 1
    IF (ABS(Z - ZNEW) .LE. TOL) GO TO 300
    IF (ITER .GT. MAXITER) THEN
        WRITE(*,*)' Max. iteration exceeded'
        WRITE(*,*)' Z, ZNEW = ',Z,ZNEW
        GO TO 300
END IF
    Z = ZNEW
    GO TO 100
                                        300 Z = ZNEW
F = 1.0 - X * X * C / R1MAG
G = DT  - X * X * X * S / SQRT(GM)
FDOT = -SQRT(GM) * X * (1.0 - Z * S) / (R1MAG * R2MAG)
GDOT = 1.0 - X * X * C / R2MAG
IF (ISWT .EQ. 0) THEN
                                        DO I = 1, 3
    V1(I) = (R2(I) - F * R1(I)) / G
    V2(I) = (GDOT * R2(I) - R1(I)) / G
    END DO
    V1MAG = SQRT(V1(1)**2 + V1(2)**2 + V1(3)**2)
                                    V2MAG = SQRT(V2(1)**2 + V2(2)**2 + V2(3)**2)
ELSE
                                    DOTP  = 2.0 * R1MAG * R2MAG * COS(ANGR)
                                V1MAG = SQRT(R2MAG**2 - F * DOTP + F*F * R1MAG**2) / G
V2MAG = SQRT(R1MAG**2 - GDOT * DOTP + GDOT*GDOT * R2MAG**2) / G
END IF
GO TO 900
                                        800 IERR = 1
900 CONTINUE
RETURN
                                            END

C     ************************************************************
FUNCTION FACT(I)
IMPLICIT REAL*8(A-H,O-Z)
VAL = 1.0
DO ITER = 1, I
                                        VAL = VAL * FLOAT(ITER)
END DO
FACT = VAL
    RETURN
```

END

References

1. Bate, R. R.,, Mueller, D. D., White, J. E., *Fundamentals of Astrodynamics*, Dover Publications, NY, 1971.
2. Battin, R. H., Vaughan, R. M., "An Elegant Lambert Algorithm," *AIAA Journal of Guidance, Control, and Dynamics*, Vol. 7, No. 6., pp. 662-670, Dec. 1984.
3. Battin, R. H., *An Introduction to the Mathematics and Methods of Astrodynamics,* AIAA Education Series, NY.
4. Nelson, S. L., Zarchan, P., "Alternative Approach to the Solution of Lambert's Problem," *AIAA Journal of Guidance, Control, and Dynamics*, Vol. 7, No. 6., pp. 662-670, Dec. 1984.
5. Vinti, J. P., *Orbital and Celestial Mechanics*, Academic Press, NY, 1993.
6. Green, D. H., and Knuth, D. E., *Mathematics for the Analysis of Algorithms*, 2nd Ed.,Birkhauser, Boston, MA, 1982.
7. Press, W. H., Teukolsky, S. A., et al, *Numerical Recipes in Fortran: The Art of Scientific Computing*, 2nd Edition, Cambridge University Press, NY, 1992.
8. Vettering, W. T., Teukolsky, S. A., et al, *Numerical Recipes: Example Book [FORTRAN]*, 2nd Edition, Cambridge University Press, NY, 1992.
9. Oikawa, S., "Radar Track Filter Initialization Study", XonTech, July 28, 1999 (PowerPoint Presentation).
10. Caampued, T., "TD/SAT Cued Search 'Hot-Start' Options," XonTech, 15 Oct. 1999 (PowerPoint Presentation).
11. O'Dea, B., "TD/SAT Modeling and Algorithms Discussion: Tracker," Xontech, 30 Sept.- 1 Oct. 1999 (Adobe Arobat Presentation).
12. Bohannon, G., "TD/SAT Modeling and Algorithms Discussion: Test Driver," Xontech, 30 Sept.- 1 Oct. 1999 (Adobe Arobat Presentation).
13. Daum, F. E., Fitzgerald, R. J., "Decoupled Kalman Filters for Phased Array Radar Tracking," *IEEE Transactions on Automatic Control,* Vol. AC-28, No. 3, pp. 264-283, March 1983.
14. Fitzgerald, R. J., "Effects of Range-Doppler Coupling on Chirp Radar Tracking Accuracy," *IEEE Transactions on Aerospace and Electronic Systems,* Vol. AES-10, No. 4, pp. 528-532, July 1974.
15. Press, W. H., Teukolsky, S. A., et al, *Numerical Recipes in Fortran 90: The Art of Parallel Scientific Computing*, Vol. 2, 2nd Edition, Cambridge University Press, NY, 1999.
16. Brown, C. D., *Spacecraft Mission Design*, AIAA Education Series, NY, 1992.

Distribution:

A. Entsminger	K. Berkowitz	R. Blanton	F. Daum
T. Parsons	R. Hettich	D. Lawrence	D. Matthieson
V. Ruggeri	T. Pham	R. Reed	D. Rypysc
S. Sillers	H. Satz	R. Seed	S. Sparagna
	F. Steudel	M. Weisman	