Least Squares and Kalman Filtering

R.E. Deakin
Bonbeach VIC, 3196, Australia
Email: randm.deakin@gmail.com

02-Sep-2015

INTRODUCTION

The theory of least squares and its application to adjustment of survey measurements is well known to every geodesist. The invention of the method is generally attributed to Carl Friedrich Gauss (1777-1855) but could equally be credited to Adrien-Marie Legendre (1752-1833). Gauss used the method of least squares to compute the elements of the orbit of the minor planet *Ceres* and predicted its position in October 1801 from a few observations made in the previous year. He published the technique in 1809 in *Theoria Motus Corporum Coelestium in Sectionibus Conicis Solem Ambientium* (Theory of the Motion of the Heavenly Bodies Moving about the Sun in Conic Sections), mentioning that he had used it since 1795, and also developed what we now know as the normal law of error, concluding that: *"... the most probable system of values of the quantities ... will be that in which the sum of the squares of the differences between the actually observed and computed values multiplied by numbers that measure the degree of precision, is a minimum."* (Gauss 1809).

Legendre published an independent development of the technique in *Nouvelles méthodes pour la détermination des orbites des comètes* (New methods for the determination of the orbits of comets), Paris, 1806 and also as the "Méthod des moindres carriés" (Method of Least Squares), published in the *Mémoires de l'Institut national des sciences at arts,* vol. 7, pt. 2, Paris, 1810.

After these initial works, the topic was subjected to rigid analysis and by the beginning of the 20th century was the universal method for the treatment of observations. Merriman (1905) compiled a list of 408 titles, including 72 books, written on the topic prior to 1877 and publication has continued unabated since then. Leahy (1974) has an excellent summary of the development of least squares and clearly identifies the historical connection with mathematical statistics, which it pre-dates.

The current literature is extensive; the books *Observations and Least Squares* (Mikhail 1976) and *Analysis and Adjustment of Survey Measurements* (Mikhail and Gracie 1981), and lecture notes by Cross (1992), Krakiwsky (1975) and Wells and Krakiwsky (1971) stand out as the simplest modern treatments of the topic.

Following Wells and Krakiwsky (1971, pp.8-9), it is interesting to analyse the following quotation from Gauss' *Theoria Motus* (Gauss, 1809, p.249).

> "If the astronomical observations and other quantities, on which the computation of orbits is based, were absolutely correct, the elements also, whether deduced from three or four observations, would be strictly accurate (so far indeed as the motion is supposed to take place exactly according to the laws of KEPLER), and, therefore, if other observations were used, they might be confirmed, but not corrected. But since all our measurements and observations are nothing more than approximations to the truth, the same must be true of all calculations resting upon them, and the highest aim of all computations made concerning concrete phenomena must be to approximate, as nearly as practicable, to the truth. But this can be accomplished in no other way than by a suitable combination of more observations than the number absolutely requisite for the determination of the unknown quantities. This problem can only be properly undertaken when an approximate knowledge of the orbit has been already attained, which is afterwards to be corrected so as to satisfy all the observations in the most accurate manner possible."

This single paragraph, written over 200 years ago, embodies the following concepts, which are as relevant today as they were then.

(i) Mathematical models may be incomplete,

(ii) Physical measurements are inconsistent,

(iii) All that can be expected from computations based on inconsistent measurements are estimates of the truth,

(iv) Redundant measurements will reduce the effect of measurement inconsistencies,

(v) Initial approximations to the final estimates should be used, and finally,

(vi) Initial approximations should be corrected in such a way as to minimise the inconsistencies between measurements (by which Gauss meant his method of least squares).

These concepts are also embedded in the Kalman Filter, an estimation process developed by Rudolf E. Kalman in 1960 (Kalman 1960). The Kalman Filter is a set of equations that can be used to determine the best estimates of a set of parameters (the state) linked to a mathematical model of a dynamic measurement system. Kalman's original development was a (linear) solution to non-linear maximum likelihood estimation; developed by R.A. Fisher[1] and studied by Kolmogorov[2] in 1942 and Weiner[3] in 1942. One of the first implementations of Kalman's filter was in the estimation of a spacecraft's trajectory and it was incorporated into the Apollo navigation computer. It is now a 'standard component' in inertial guidance systems. Kalman filtering is also used in kinematic GPS and most modern navigation systems.

A Kalman Filter can be thought of as a logical extension of Gauss' original development of least squares to estimate unknown parameters of a system. In Gauss' era and up until the middle 20th century, systems (whose parameters were to be estimated) were generally static and measurements unvarying with respect to time. But in this modern era, systems may be dynamic and measurements made from moving platforms at regular time intervals. The estimation process must link these measurements and the Kalman filter achieves this; estimating the state of a system (the parameters) at intervals of time.

These notes contain derivations of formula and worked examples of least squares estimation (including Kalman filtering). First, there is a general treatment of least squares estimation that is called here *Combined Least Squares* which can be shown to encompass the usual solutions known in surveying and geodesy as adjustment of indirect observations and adjustment of observations only. This is followed by the derivation of the Kalman Filter equations using the same basic principles – minimizing sum of squares of weighted residuals. The derivations are concise and the interested reader is directed to more extensive developments as references.

[1] Sir Ronald Aylmer Fisher (1890–1962) was an English statistician and mathematician known for his important contributions to statistics, including the analysis of variance, maximum likelihood, fiducial inference, and the derivation of various sampling distributions.

[2] A.N. Kolmogorov (1903–1987) was a 20th-century Russian mathematician who made significant contributions to the mathematics of probability theory, topology, classical mechanics and information theory.

[3] Norbert Wiener (1894–1964) was an American child prodigy who became a mathematician and philosopher and Professor of Mathematics at MIT. Wiener was an early researcher in stochastic and noise processes, contributing work relevant to electronic engineering, electronic communication, and control systems.

COMBINED LEAST SQUARES

A common treatment of the least squares technique of estimation starts with simple linear mathematical models having observations (or measurements) as explicit functions of parameters with non-linear models developed as extensions. This adjustment technique is generally described as <u>adjustment of indirect observations</u> (also called parametric least squares). Cases where the mathematical models contain only measurements are usually treated separately and this technique is often described as <u>adjustment of observations only</u> (also called condition equations). Both techniques are of course particular cases of a general technique called here *Combined Least Squares,* the solution of which is set out below. This general technique also assumes that the parameters, if any, can be treated as 'observables', i.e., they have an a priori covariance matrix. This concept allows the general technique to be adapted to sequential processing of data where parameters are updated by the addition of new observations.

In general, least squares solutions require iteration, since a non-linear model is assumed. The iterative process is explained below. In addition, a proper treatment of covariance propagation is presented and cofactor matrices given for all the computed and derived quantities in the adjustment process. Finally, the particular cases of the general least squares technique are described.

Consider the following set of non-linear equations representing the mathematical model in an adjustment

$$F\left(\hat{\mathbf{l}}, \hat{\mathbf{x}}\right) = \mathbf{0} \tag{1}$$

where $\mathbf{l}$ is a vector of n observations and $\mathbf{x}$ is a vector of u parameters; $\hat{\mathbf{l}}$ and $\hat{\mathbf{x}}$ referring to estimates derived from the least squares process such that

$$\hat{\mathbf{l}} = \mathbf{l} + \mathbf{v} \quad \text{and} \quad \hat{\mathbf{x}} = \mathbf{x} + \delta\mathbf{x} \tag{2}$$

where $\mathbf{v}$ is a vector of residuals or small corrections and $\delta\mathbf{x}$ is a vector of small corrections. The observations $\mathbf{l}$ have an a priori cofactor matrix $\mathbf{Q}_{ll}$ containing estimates of the variances and covariances of the observations. In many cases the observations are independent and $\mathbf{Q}_{ll}$ is diagonal. In this general technique, the parameters $\mathbf{x}$ are treated as 'observables' with a full a priori cofactor matrix $\mathbf{Q}_{xx}$. The diagonal elements of $\mathbf{Q}_{xx}$ contain estimates of variances of the parameters and the off-diagonal elements contain estimates of the covariances between parameters. Cofactor matrices $\mathbf{Q}_{ll}$ and $\mathbf{Q}_{xx}$ are related to the covariance matrices $\mathbf{\Sigma}_{ll}$ and $\mathbf{\Sigma}_{xx}$ by the variance factor σ_0^2

$$\mathbf{\Sigma}_{ll} = \sigma_0^2\mathbf{Q}_{ll} \quad \mathbf{\Sigma}_{xx} = \sigma_0^2\mathbf{Q}_{xx} \tag{3}$$

Also, weight matrices $\mathbf{W}$ are useful and are defined, in general, as the inverse of cofactor matrices

$$\mathbf{W} = \mathbf{Q}^{-1} \tag{4}$$

and covariance, cofactor and weight matrices are all symmetric, hence $\mathbf{Q}^T = \mathbf{Q}$ and $\mathbf{W}^T = \mathbf{W}$ where the superscript T denotes the transpose of the matrix.

Note also, that in this development where $\mathbf{Q}$ and $\mathbf{W}$ are written without subscripts they refer to the observations, i.e., $\mathbf{Q}_{ll} = \mathbf{Q}$ and $\mathbf{W}_{ll} = \mathbf{W}$

Linearizing (1) using Taylor's theorem and ignoring 2nd and higher-order terms, gives

$$F\left(\hat{\mathbf{l}},\hat{\mathbf{x}}\right) = F\left(\mathbf{l},\mathbf{x}\right) + \left.\frac{\partial F}{\partial \hat{\mathbf{l}}}\right|_{l,x}\left(\hat{\mathbf{l}}-\mathbf{l}\right) + \left.\frac{\partial F}{\partial \hat{\mathbf{x}}}\right|_{l,x}\left(\hat{\mathbf{x}}-\mathbf{x}\right) = \mathbf{0} \tag{5}$$

and with $\mathbf{v} = \hat{\mathbf{l}} - \mathbf{l}$ and $\delta\mathbf{x} = \hat{\mathbf{x}} - \mathbf{x}$ from (2) we may write the linearized model in symbolic form as

$$\mathbf{Av} + \mathbf{B}\,\delta\mathbf{x} = \mathbf{f} \tag{6}$$

Equation (6) represents a system of *m* equations that will be used to estimate the *u* parameters from *n* observations. It is assumed that this is a redundant system where

$$n \geq m \geq u \tag{7}$$

and

$$r = m - u \tag{8}$$

is the redundancy or degrees of freedom.

In equation (6) the coefficient matrices **A** and **B** are design matrices containing partial derivatives of the function evaluated using the observations **l** and the "observed" parameters **x**.

$$\mathbf{A}_{m,n} = \left.\frac{\partial F}{\partial \hat{\mathbf{l}}}\right|_{l,x} \qquad \mathbf{B}_{m,u} = \left.\frac{\partial F}{\partial \hat{\mathbf{x}}}\right|_{l,x} \tag{9}$$

The vector **f** contains *m* numeric terms calculated from the functional model using **l** and **x**.

$$\mathbf{f}_{m,1} = -\left\{F\left(\mathbf{l},\mathbf{x}\right)\right\} \tag{10}$$

THE COMBINED LEAST SQUARES SOLUTION

The least squares solution of (6), i.e., the solution which makes the sums of the squares of the weighted residuals a minimum, is obtained by minimizing the scalar function φ

$$\varphi = \mathbf{v}^T\mathbf{W}\,\mathbf{v} + \delta\mathbf{x}^T\mathbf{W}_{xx}\delta\mathbf{x} - 2\mathbf{k}^T\left(\mathbf{Av} + \mathbf{B}\delta\mathbf{x} - \mathbf{f}\right) \tag{11}$$

where **k** is a vector of *m* Lagrange multipliers which are at this stage unknown and the 2 is added for convenience later on.

φ is a minimum when its derivatives with respect to **v** and $\delta\mathbf{x}$ are equated to zero, i.e.

$$\frac{\partial\varphi}{\partial\mathbf{v}} = 2\mathbf{v}^T\mathbf{W} - 2\mathbf{k}^T\mathbf{A} = \mathbf{0}^T \qquad \text{and} \qquad \frac{\partial\varphi}{\partial\delta\mathbf{x}} = 2\delta\mathbf{x}^T\mathbf{W}_{xx} - 2\mathbf{k}^T\mathbf{B} = \mathbf{0}^T$$

These equations can be simplified by dividing both sides by two, transposing and changing signs to give

$$-\mathbf{Wv} + \mathbf{A}^T\mathbf{k} = \mathbf{0} \qquad \text{and} \qquad -\mathbf{W}_{xx}\delta\mathbf{x} + \mathbf{B}^T\mathbf{k} = \mathbf{0} \tag{12}$$

Equations (12) can be combined with (6) and arranged in matrix form as

$$\begin{bmatrix} -\mathbf{W} & \mathbf{A}^T & \mathbf{0} \\ \mathbf{A} & \mathbf{0} & \mathbf{B} \\ \mathbf{0} & \mathbf{B}^T & -\mathbf{W}_{xx} \end{bmatrix} \begin{bmatrix} \mathbf{v} \\ \mathbf{k} \\ \delta\mathbf{x} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{f} \\ \mathbf{0} \end{bmatrix} \tag{13}$$

Equation (13) can be solved by the following reduction process given by Cross (1992, pp. 22-23). Consider the partitioned matrix equation $\mathbf{P}\,\mathbf{y} = \mathbf{u}$ given as

$$\left[\begin{array}{c|c} \mathbf{P}_{11} & \mathbf{P}_{12} \\ \hline \mathbf{P}_{21} & \mathbf{P}_{22} \end{array}\right] \left[\begin{array}{c} \mathbf{y}_1 \\ \hline \mathbf{y}_2 \end{array}\right] = \left[\begin{array}{c} \mathbf{u}_1 \\ \hline \mathbf{u}_2 \end{array}\right] \tag{14}$$

which can be expanded to give

$$\mathbf{P}_{11}\mathbf{y}_1 + \mathbf{P}_{12}\mathbf{y}_2 = \mathbf{u}_1$$

or

$$\mathbf{y}_1 = \mathbf{P}_{11}^{-1}\left(\mathbf{u}_1 - \mathbf{P}_{12}\mathbf{y}_2\right) \tag{15}$$

Eliminating $\mathbf{y}_1$ by substituting (15) into (14) gives

$$\left[\begin{array}{c|c} \mathbf{P}_{11} & \mathbf{P}_{12} \\ \hline \mathbf{P}_{21} & \mathbf{P}_{22} \end{array}\right] \left[\begin{array}{c} \mathbf{P}_{11}^{-1}\left(\mathbf{u}_1 - \mathbf{P}_{12}\mathbf{y}_2\right) \\ \hline \mathbf{y}_2 \end{array}\right] = \left[\begin{array}{c} \mathbf{u}_1 \\ \hline \mathbf{u}_2 \end{array}\right]$$

Expanding the matrix equation gives

$$\begin{aligned} \mathbf{P}_{21}\mathbf{P}_{11}^{-1}\left(\mathbf{u}_1 - \mathbf{P}_{12}\mathbf{y}_2\right) + \mathbf{P}_{22}\mathbf{y}_2 &= \mathbf{u}_2 \\ \mathbf{P}_{21}\mathbf{P}_{11}^{-1}\mathbf{u}_1 - \mathbf{P}_{21}\mathbf{P}_{11}^{-1}\mathbf{P}_{12}\mathbf{y}_2 + \mathbf{P}_{22}\mathbf{y}_2 &= \mathbf{u}_2 \end{aligned}$$

and an expression for $\mathbf{y}_2$ is given by

$$\left(\mathbf{P}_{22} - \mathbf{P}_{21}\mathbf{P}_{11}^{-1}\mathbf{P}_{12}\right)\mathbf{y}_2 = \mathbf{u}_2 - \mathbf{P}_{21}\mathbf{P}_{11}^{-1}\mathbf{u}_1 \tag{16}$$

Now partitioning (13) in the same way as (14) gives

$$\left[\begin{array}{c|cc} -\mathbf{W} & \mathbf{A}^T & \mathbf{0} \\ \hline \mathbf{A} & \mathbf{0} & \mathbf{B} \\ \mathbf{0} & \mathbf{B}^T & -\mathbf{W}_{xx} \end{array}\right] \left[\begin{array}{c} \mathbf{v} \\ \hline \mathbf{k} \\ \delta\mathbf{x} \end{array}\right] = \left[\begin{array}{c} \mathbf{0} \\ \hline \mathbf{f} \\ \mathbf{0} \end{array}\right] \tag{17}$$

and then eliminating **v** by applying (16) yields

$$\left[\begin{bmatrix} \mathbf{0} & \mathbf{B} \\ \mathbf{B}^T & -\mathbf{W}_{xx} \end{bmatrix} - \begin{bmatrix} \mathbf{A} \\ \mathbf{0} \end{bmatrix}\left[-\mathbf{W}^{-1}\right]\left[\mathbf{A}^T \quad \mathbf{0}\right]\right] \begin{bmatrix} \mathbf{k} \\ \delta\mathbf{x} \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ \mathbf{0} \end{bmatrix} - \begin{bmatrix} \mathbf{A} \\ \mathbf{0} \end{bmatrix}\left[-\mathbf{W}^{-1}\right][\mathbf{0}]$$

Remembering that $\mathbf{Q} = \mathbf{W}^{-1}$ the equation can be simplified as

$$\left[\begin{array}{c|c} \mathbf{AQA}^T & \mathbf{B} \\ \hline \mathbf{B}^T & -\mathbf{W}_{xx} \end{array}\right] \left[\begin{array}{c} \mathbf{k} \\ \hline \delta\mathbf{x} \end{array}\right] = \left[\begin{array}{c} \mathbf{f} \\ \hline \mathbf{0} \end{array}\right] \tag{18}$$

Again, applying (16) to the partitioned equation (18) gives

$$\left(-\mathbf{W}_{xx}-\mathbf{B}^T\left(\mathbf{AQA}^T\right)^{-1}\mathbf{B}\right)\delta\mathbf{x}=\mathbf{0}-\mathbf{B}^T\left(\mathbf{AQA}^T\right)^{-1}\mathbf{f}$$

and re-arranging gives the normal equations

$$\left(\mathbf{B}^T\left(\mathbf{AQA}^T\right)^{-1}\mathbf{B}+\mathbf{W}_{xx}\right)\delta\mathbf{x}=\mathbf{B}^T\left(\mathbf{AQA}^T\right)^{-1}\mathbf{f} \tag{19}$$

Mikhail (1976, p. 114) simplifies (19) by introducing *equivalent observations* $\mathbf{l}_e$ where

$$\mathbf{l}_e=\mathbf{A}\,\mathbf{l} \tag{20}$$

Applying the matrix rule for cofactor propagation (Mikhail 1976, pp. 76-79) gives the cofactor matrix of the equivalent observations as

$$\mathbf{Q}_e=\mathbf{AQA}^T \tag{21}$$

With the usual relationship between weight matrices and cofactor matrices, [see (4)], we may write

$$\mathbf{W}_e=\mathbf{Q}_e^{-1}=\left(\mathbf{AQA}^T\right)^{-1} \tag{22}$$

Using (22) in (19) gives the normal equations as

$$\left(\mathbf{B}^T\mathbf{W}_e\mathbf{B}+\mathbf{W}_{xx}\right)\delta\mathbf{x}=\mathbf{B}^T\mathbf{W}_e\mathbf{f} \tag{23}$$

with the auxiliaries **N** and **t** as

$$\mathbf{N}=\mathbf{B}^T\mathbf{W}_e\mathbf{B}\qquad\mathbf{t}=\mathbf{B}^T\mathbf{W}_e\mathbf{f} \tag{24}$$

The vector of corrections $\delta\mathbf{x}$ is given by

$$\delta\mathbf{x}=\left(\mathbf{N}+\mathbf{W}_{xx}\right)^{-1}\mathbf{t} \tag{25}$$

The reduction process applied to (13) also yields the vector of Lagrange multipliers **k**

$$\mathbf{k}=\left(\mathbf{AQA}^T\right)^{-1}\left(\mathbf{f}-\mathbf{B}\,\delta\mathbf{x}\right)=\mathbf{W}_e\left(\mathbf{f}-\mathbf{B}\,\delta\mathbf{x}\right) \tag{26}$$

and the vector of residuals **v** is obtained from (12) as

$$\mathbf{v}=\mathbf{W}^{-1}\mathbf{A}^T\mathbf{k}=\mathbf{QA}^T\mathbf{k} \tag{27}$$

THE ITERATIVE PROCESS OF SOLUTION

Remembering that $\hat{\mathbf{x}}=\mathbf{x}+\delta\mathbf{x}$, where **x** is the vector of a priori estimates of the parameters, $\delta\mathbf{x}$ is a vector of corrections and $\hat{\mathbf{x}}$ is the least squares estimate of the parameters.

At the beginning of the iterative solution, it can be assumed that $\hat{\mathbf{x}}$ equals the a priori estimates $\mathbf{x}_1$ and a set of corrections $\delta\mathbf{x}_1$ computed. These are added to $\mathbf{x}_1$ giving an updated set $\mathbf{x}_2$. **A** and **B** are recalculated and a new weight matrix $\mathbf{W}_{xx}$ computed by cofactor propagation.

The corrections are computed again, and the whole process cycles through until the corrections reach some predetermined value, which terminates the process.

$$\hat{\mathbf{x}}_{n+1} = \mathbf{x}_n + \delta\mathbf{x}_n \tag{28}$$

COFACTOR MATRICES

Derivation of the cofactor matrices is a lengthy process and the results given below can be found in Mikhail (1976, pp. 349-359)

Cofactor Matrix for $\hat{\mathbf{x}}$ $$\mathbf{Q}_{\hat{x}\hat{x}} = \left(\mathbf{N} + \mathbf{W}_{xx}\right)^{-1} \tag{29}$$

Cofactor Matrix for $\hat{\mathbf{l}}$ $$\mathbf{Q}_{\hat{l}\hat{l}} = \mathbf{Q} + \mathbf{Q}\mathbf{A}^T\mathbf{W}_e\mathbf{B}\left(\mathbf{N} + \mathbf{W}_{xx}\right)^{-1}\mathbf{B}^T\mathbf{W}_e\mathbf{A}\mathbf{Q} - \mathbf{Q}\mathbf{A}^T\mathbf{W}_e\mathbf{A}\mathbf{Q} \tag{30}$$

Cofactor Matrix for $\delta\mathbf{x}$ $$\mathbf{Q}_{\delta\mathbf{x}\delta\mathbf{x}} = \left(\mathbf{N} + \mathbf{W}_{xx}\right)^{-1}\mathbf{N}\mathbf{Q}_{xx} \tag{31}$$

Cofactor Matrix for $\mathbf{v}$ $$\mathbf{Q}_{vv} = \mathbf{Q} - \mathbf{Q}_{\hat{l}\hat{l}} \tag{32}$$

Covariance Matrix $\mathbf{\Sigma}_{\hat{x}\hat{x}}$ $$\mathbf{\Sigma}_{\hat{x}\hat{x}} = \sigma_0^2\,\mathbf{Q}_{\hat{x}\hat{x}} \tag{33}$$

The estimated variance factor is $$\sigma_0^2 = \frac{\mathbf{v}^T\mathbf{W}\mathbf{v} + \delta\mathbf{x}^T\mathbf{W}_{xx}\delta\mathbf{x}}{r} \tag{34}$$

where the degrees of freedom *r* are $$r = m - u + u_x \tag{35}$$

m is the number of equations used to estimate the *u* parameters from *n* observations. u_x is the number of weighted parameters. [(35) is given by Krakiwsky (1975, p.17, eq. 2-62) who notes that it is an approximation only and directs the reader to Bossler (1972) for a complete and rigorous treatment.]

GENERATION OF THE STANDARD LEAST SQUARES CASES

Combined Case with Weighted Parameters $\left(\mathbf{A};\ \mathbf{B};\ \mathbf{W};\ \mathbf{W}_{xx}\right)$

$$\mathbf{A}\mathbf{v} + \mathbf{B}\delta\mathbf{x} = \mathbf{f} \quad \text{with} \quad \mathbf{W} = \mathbf{Q}^{-1} \text{ and } \mathbf{W}_{xx} \neq \mathbf{0}$$

The general case of a non-linear implicit model with weighted parameters treated as observables is known as the Combined Case with Weighted Parameters. It has a solution given by the following equations.

$$\delta\mathbf{x} = \left(\mathbf{N} + \mathbf{W}_{xx}\right)^{-1}\mathbf{t} \tag{36}$$

with $$\mathbf{N} = \mathbf{B}^T\mathbf{W}_e\mathbf{B} \qquad \mathbf{t} = \mathbf{B}^T\mathbf{W}_e\mathbf{f} \qquad \mathbf{W}_e = \mathbf{Q}_e^{-1} = \left(\mathbf{A}\mathbf{Q}\mathbf{A}^T\right)^{-1} \tag{37}$$

$$\hat{\mathbf{x}} = \mathbf{x} + \delta\mathbf{x} \tag{38}$$

$$\mathbf{k} = \mathbf{W}_e\left(\mathbf{f} - \mathbf{B}\,\delta\mathbf{x}\right) \tag{39}$$

$$\mathbf{v} = \mathbf{W}^{-1}\mathbf{A}^T\mathbf{k} = \mathbf{Q}\mathbf{A}^T\mathbf{k} \tag{40}$$

$$\hat{\mathbf{l}} = \mathbf{l} + \mathbf{v} \tag{41}$$

$$\mathbf{Q}_{\hat{x}\hat{x}} = \left(\mathbf{N} + \mathbf{W}_{xx}\right)^{-1} \tag{42}$$

$$\mathbf{Q}_{\hat{l}\hat{l}} = \mathbf{Q} + \mathbf{Q}\mathbf{A}^T\mathbf{W}_e\mathbf{B}\left(\mathbf{N} + \mathbf{W}_{xx}\right)^{-1}\mathbf{B}^T\mathbf{W}_e\mathbf{A}\mathbf{Q} - \mathbf{Q}\mathbf{A}^T\mathbf{W}_e\mathbf{A}\mathbf{Q} \tag{43}$$

$$\mathbf{Q}_{vv} = \mathbf{Q} - \mathbf{Q}_{\hat{l}\hat{l}} \tag{44}$$

$$\sigma_0^2 = \frac{\mathbf{v}^T\mathbf{W}\mathbf{v} + \delta\mathbf{x}^T\mathbf{W}_{xx}\delta\mathbf{x}}{r} = \frac{\mathbf{v}^T\mathbf{W}\mathbf{v} + \delta\mathbf{x}^T\mathbf{W}_{xx}\delta\mathbf{x}}{m - u + u_x} \tag{45}$$

$$\mathbf{\Sigma}_{\hat{x}\hat{x}} = \sigma_0^2\,\mathbf{Q}_{\hat{x}\hat{x}} \quad \mathbf{\Sigma}_{vv} = \sigma_0^2\,\mathbf{Q}_{vv} \quad \mathbf{\Sigma}_{\hat{l}\hat{l}} = \sigma_0^2\,\mathbf{Q}_{\hat{l}\hat{l}} \tag{46}$$

Combined Case with $\mathbf{A}$; $\mathbf{B}$; $\mathbf{W}$; $\mathbf{W}_{xx} = \mathbf{0}$

$$\mathbf{A}\mathbf{v} + \mathbf{B}\delta\mathbf{x} = \mathbf{f} \text{ with } \mathbf{W} = \mathbf{Q}^{-1} \text{ and } \mathbf{W}_{xx} = \mathbf{0}$$

The Combined Case is a non-linear implicit mathematical model with no weights on the parameters. The set of equations for the solution is deduced from the Combined Case with Weighted Parameters by considering that if there are no weights then $\mathbf{W}_{xx} = \mathbf{0}$ and $\mathbf{Q}_{xx} = \mathbf{0}$. This implies that **x** is a constant vector (denoted by $\mathbf{x}^0$) of approximate values of the parameters, and partial derivatives with respect to $\mathbf{x}^0$ are undefined. Substituting these two null matrices and the constant vector $\mathbf{x} = \mathbf{x}^0$ into equations (36) to (45) gives the following results.

$$\delta\mathbf{x} = \mathbf{N}^{-1}\mathbf{t} \tag{47}$$

with $\quad \mathbf{N} = \mathbf{B}^T\mathbf{W}_e\mathbf{B} \quad \mathbf{t} = \mathbf{B}^T\mathbf{W}_e\mathbf{f}^0 \quad \mathbf{f} = -F\left(\mathbf{x}^0, \mathbf{l}\right) \quad \mathbf{W}_e = \mathbf{Q}_e^{-1} = \left(\mathbf{A}\mathbf{Q}\mathbf{A}^T\right)^{-1} \quad (48)$

$$\hat{\mathbf{x}} = \mathbf{x}^0 + \delta\mathbf{x} \tag{49}$$

$$\mathbf{k} = \mathbf{W}_e\left(\mathbf{f} - \mathbf{B}\,\delta\mathbf{x}\right) \tag{50}$$

$$\mathbf{v} = \mathbf{W}^{-1}\mathbf{A}^T\mathbf{k} = \mathbf{Q}\mathbf{A}^T\mathbf{k} \tag{51}$$

$$\hat{\mathbf{l}} = \mathbf{l} + \mathbf{v} \tag{52}$$

$$\mathbf{Q}_{\delta\mathbf{x}\delta\mathbf{x}} = \mathbf{Q}_{\hat{x}\hat{x}} = \mathbf{N}^{-1} \tag{53}$$

$$\mathbf{Q}_{\hat{l}\hat{l}} = \mathbf{Q} + \mathbf{Q}\mathbf{A}^T\mathbf{W}_e\mathbf{B}\,\mathbf{N}^{-1}\mathbf{B}^T\mathbf{W}_e\mathbf{A}\mathbf{Q} - \mathbf{Q}\mathbf{A}^T\mathbf{W}_e\mathbf{A}\mathbf{Q} \tag{54}$$

$$\mathbf{Q}_{vv} = \mathbf{Q} - \mathbf{Q}_{\hat{l}\hat{l}} \tag{55}$$

$$\sigma_0^2 = \frac{\mathbf{v}^T\mathbf{W}\mathbf{v}}{r} = \frac{\mathbf{v}^T\mathbf{W}\mathbf{v}}{m-u} \tag{56}$$

$$\mathbf{\Sigma}_{\hat{x}\hat{x}} = \sigma_0^2\,\mathbf{Q}_{\hat{x}\hat{x}} \quad \mathbf{\Sigma}_{vv} = \sigma_0^2\,\mathbf{Q}_{vv} \quad \mathbf{\Sigma}_{\hat{l}\hat{l}} = \sigma_0^2\,\mathbf{Q}_{\hat{l}\hat{l}} \tag{57}$$

Indirect Least Squares (Parametric Case) with $\mathbf{A} = \mathbf{I}$; $\mathbf{B}$; $\mathbf{W}$; $\mathbf{W}_{xx} = \mathbf{0}$

$$\mathbf{v} + \mathbf{B}\delta\mathbf{x} = \mathbf{f} \;\text{ with }\; \mathbf{W} = \mathbf{Q}^{-1} \text{ and } \mathbf{W}_{xx} = \mathbf{0}$$

Indirect Least Squares (Parametric Case) is a mathematical model with the observations **l** explicitly expressed by some non-linear function of the parameters **x** only. This implies that the design matrix **A** is equal to the identity matrix **I**. Setting $\mathbf{A} = \mathbf{I}$ in the Combined Case (with no weights) leads to the following equations.

$$\delta\mathbf{x} = \mathbf{N}^{-1}\mathbf{t} \tag{58}$$

with

$$\mathbf{N} = \mathbf{B}^T\mathbf{W}\mathbf{B} \qquad \mathbf{t} = \mathbf{B}^T\mathbf{W}\mathbf{f}^0 \qquad \mathbf{f} = -F\left(\mathbf{x}^0, \mathbf{l}\right) \tag{59}$$

$$\hat{\mathbf{x}} = \mathbf{x}^0 + \delta\mathbf{x} \tag{60}$$

$$\mathbf{v} = \mathbf{f} - \mathbf{B}\,\delta\mathbf{x} \tag{61}$$

$$\hat{\mathbf{l}} = \mathbf{l} + \mathbf{v} \tag{62}$$

$$\mathbf{Q}_{\delta\mathbf{x}\delta\mathbf{x}} = \mathbf{Q}_{\hat{x}\hat{x}} = \mathbf{N}^{-1} \tag{63}$$

$$\mathbf{Q}_{vv} = \mathbf{Q} - \mathbf{B}\mathbf{N}^{-1}\mathbf{B}^T \tag{64}$$

$$\mathbf{Q}_{\hat{l}\hat{l}} = \mathbf{B}\,\mathbf{N}^{-1}\mathbf{B}^T \tag{65}$$

$$\sigma_0^2 = \frac{\mathbf{v}^T\mathbf{W}\mathbf{v}}{r} = \frac{\mathbf{v}^T\mathbf{W}\mathbf{v}}{n-u} \tag{66}$$

$$\mathbf{\Sigma}_{\hat{x}\hat{x}} = \sigma_0^2\,\mathbf{Q}_{\hat{x}\hat{x}} \quad \mathbf{\Sigma}_{vv} = \sigma_0^2\,\mathbf{Q}_{vv} \quad \mathbf{\Sigma}_{\hat{l}\hat{l}} = \sigma_0^2\,\mathbf{Q}_{\hat{l}\hat{l}} \tag{67}$$

Observations Only Least Squares (Condition Case) with $\mathbf{A}$; $\mathbf{B}=\mathbf{0}$; $\mathbf{W}$; $\mathbf{W}_{xx}=\mathbf{0}$

$$\mathbf{Av}=\mathbf{f} \quad \text{with} \quad \mathbf{W}=\mathbf{Q}^{-1} \text{ and } \mathbf{W}_{xx} \neq \mathbf{0}$$

Observations Only Least Squares (Condition Case) is characterized by a non-linear model consisting of observations only. Setting $\mathbf{B}=\mathbf{0}$ in the Combined Case (with no weights) leads to the following equations.

$$\mathbf{k}=\mathbf{W}_e\mathbf{f} \tag{68}$$

with

$$\mathbf{W}_e=\mathbf{Q}_e^{-1}=\left(\mathbf{AQA}^T\right)^{-1} \qquad \mathbf{f}=-F\left(\mathbf{l}\right) \tag{69}$$

$$\mathbf{v}=\mathbf{W}^{-1}\mathbf{A}^T\mathbf{k}=\mathbf{QA}^T\mathbf{k} \tag{70}$$

$$\hat{\mathbf{l}}=\mathbf{l}+\mathbf{v} \tag{71}$$

$$\mathbf{Q}_{\hat{l}\hat{l}}=\mathbf{Q}-\mathbf{QA}^T\mathbf{W}_e\mathbf{AQ} \tag{72}$$

$$\mathbf{Q}_{vv}=\mathbf{Q}-\mathbf{Q}_{\hat{l}\hat{l}} \tag{73}$$

$$\sigma_0^2=\frac{\mathbf{v}^T\mathbf{Wv}}{r}=\frac{\mathbf{v}^T\mathbf{Wv}}{m} \tag{74}$$

$$\mathbf{\Sigma}_{vv}=\sigma_0^2\mathbf{Q}_{vv} \qquad \mathbf{\Sigma}_{\hat{l}\hat{l}}=\sigma_0^2\mathbf{Q}_{\hat{l}\hat{l}} \tag{75}$$

EXAMPLES

The following simple examples of least squares solutions (or least squares estimations) show how appropriate mathematical models are developed and systems of (matrix) equations solved to give estimates of parameters and/or residuals. MATLAB functions are given in the Appendix and show how these solutions may be programmed.

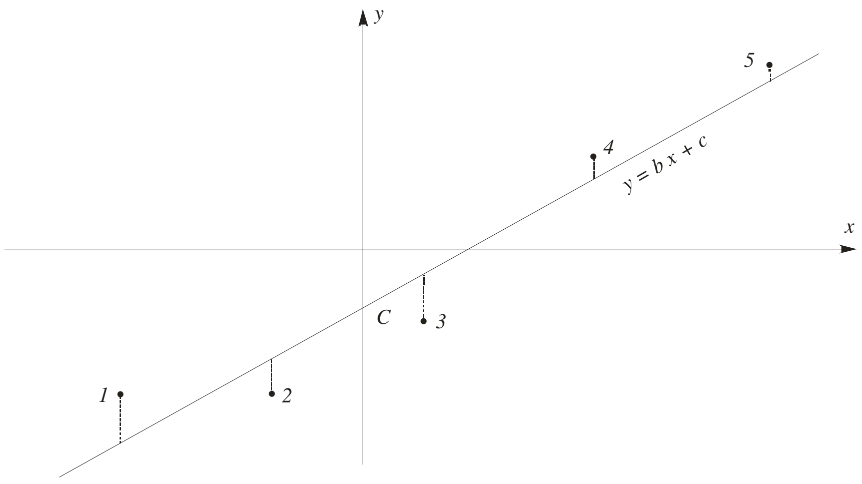
Examples 1 and 2 are both fitting straight lines $y=bx+c$ through data points. This is also known as linear regression. Example 1 assumes that the x-values are error free and the y-values are the measurements (subject to error) with associated residuals and weights reflecting the precision. The least squares solution for parameters b (slope) and c (y-intercept) is obtained using Indirect Least Squares (Parametric case). Example 2 assumes that both the x and y-values are measurements (with associated residuals) with estimates of precision (variances and covariances). The least squares solution for the parameters b and c uses Combined Least Squares. This technique is rarely explained in textbooks on the topic.

Example 3 is the solution of a small level network using two methods; Observations Only Least Squares (Condition case) and Indirect Least Squares (Parametric case)

Example 4 is position fix from measured distances to beacons of known coordinates using Indirect

Least Squares

Example 1: Line of Best Fit



Point	x (mm)	y (mm)	weight w
1	−40.0	−24.0	2
2	−15.0	−24.0	5
3	10.0	−12.0	7
4	38.0	15.0	3
5	67.0	30.0	3

Figure 1. Line of Best Fit through data points 1 to 5

The line of best fit shown in the Figure 1 has the equation $y = bx + c$ where b is the slope of the line $\left(b = \tan\theta = \dfrac{y_2 - y_1}{x_2 - x_1} \right)$ and c is the intercept of the line on the y axis.

b and c are the parameters and the data points are assumed to accord with the mathematical model $y = bx + c$ and the x,y coordinate pairs of each data point are considered as indirect measurements of the parameters m and c of the mathematical model. The column of weights reflects the differing precision associated with the measured y-values (large weight equals small precision) and the x-values are considered to be error free.

Now, since the x-values are error-free, the residuals v are associated with the measured y-values only, which leads to an observation equation of the form

$$y_k + v_k = b x_k + c \tag{76}$$

This equation can be re-arranged into a form where the 'unknowns' (v_k, b, c) are on the left-hand side of the equals sign and the 'knowns' (y_k) are on the right-hand side

$$v_k - b x_k - c = y_k \tag{77}$$

For the 5 data points there are n = 5 equations in u = 2 parameters

$$\begin{aligned} v_1 - x_1 b - c &= -y_1 \\ v_2 - x_2 b - c &= -y_2 \\ v_3 - x_3 b - c &= -y_3 \\ v_4 - x_4 b - c &= -y_4 \\ v_5 - x_5 b - c &= -y_5 \end{aligned} \tag{78}$$

These can be written in the matrix form $\mathbf{v}_{(n,1)} + \mathbf{B}_{(n,u)}\mathbf{x}_{(u,1)} = \mathbf{f}_{(n,1)}$ (Parametric Case: $\mathbf{A} = \mathbf{I}$; $\mathbf{B}$; $\mathbf{W}; \mathbf{W}_{xx} = \mathbf{0}$) as

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{bmatrix} + \begin{bmatrix} -x_1 & -1 \\ -x_2 & -1 \\ -x_3 & -1 \\ -x_4 & -1 \\ -x_5 & -1 \end{bmatrix} \begin{bmatrix} b \\ c \end{bmatrix} = \begin{bmatrix} -y_1 \\ -y_2 \\ -y_3 \\ -y_4 \\ -y_5 \end{bmatrix} \tag{79}$$

The numerical values in the matrices **B**, **f** and **W** are

$$\mathbf{B}_{(n,u)} = \begin{bmatrix} 40 & -1 \\ 15 & -1 \\ -10 & -1 \\ -38 & -1 \\ -67 & -1 \end{bmatrix} \quad \mathbf{f}_{(n,1)} = \begin{bmatrix} 24 \\ 24 \\ 12 \\ -15 \\ -30 \end{bmatrix} \quad \mathbf{W}_{(n,n)} = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 & 0 \\ 0 & 0 & 7 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 3 \end{bmatrix}$$

The solution is given by equations (58) to (67). In this particular case, the solution for the vector **x** (containing the parameters b and c) is direct; no iteration and no approximate values required.

The solution is given by

$$\mathbf{N}_{(u,u)} = \mathbf{B}^T_{(u,n)}\mathbf{W}_{(n,n)}\mathbf{B}_{(n,u)} = \begin{bmatrix} 22824.00 & 230.00 \\ 230.00 & 20.00 \end{bmatrix} \quad \mathbf{t}_{(u,1)} = \mathbf{B}^T_{(u,n)}\mathbf{W}_{(n,n)}\mathbf{f}_{(n,1)} = \begin{bmatrix} 10620.00 \\ -117.00 \end{bmatrix}$$

and $$\mathbf{x}_{(u,1)} = \begin{bmatrix} b \\ c \end{bmatrix} = \mathbf{N}^{-1}_{(u,u)}\mathbf{t}_{(u,1)} = \begin{bmatrix} 4.9556\text{e-}05 & -5.6990\text{e-}04 \\ -5.6990\text{e-}04 & 5.6554\text{e-}02 \end{bmatrix} \begin{bmatrix} 10620.00 \\ -117.00 \end{bmatrix} = \begin{bmatrix} 0.592968 \\ -12.669131 \end{bmatrix}$$

The slope of the line $b = \tan\theta = 0.592968$ and $\theta = 30^\circ\, 40'00''$ measured anticlockwise from the x-axis and the line cuts the y-axis at -12.669131

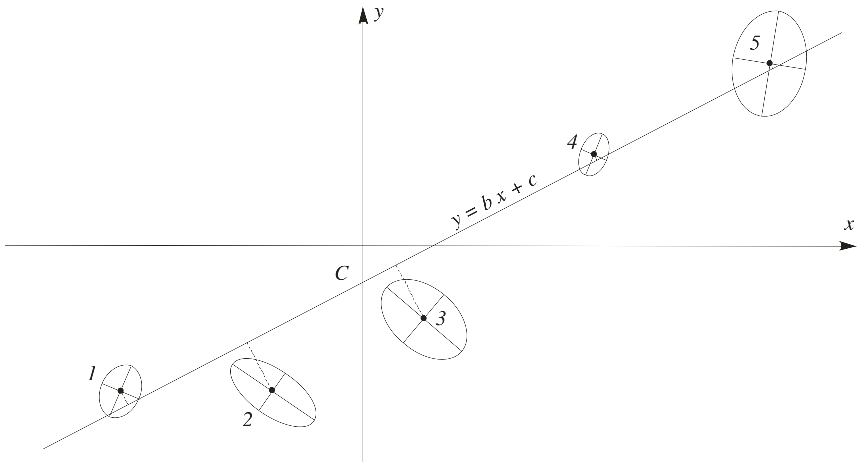
The residuals (mm) are $$\mathbf{v}_{(n,1)} = \begin{bmatrix} -12.387849 \\ 2.436350 \\ 5.260548 \\ -5.136350 \\ -2.940279 \end{bmatrix}$$

A MATLAB program *least_squares.m* is given in the Appendix and can be used to solve parametric least squares problems given the coefficient matrix **B**, the vector of numeric terms **f** and a vector of weights **w** that are the elements of a diagonal matrix **W**. This program requires a data file (an ASCII text file) with an extension .txt containing the elements of **B**, **f** and **w**.

The output from the program containing estimates of parameters and residuals and relevant cofactor matrices is placed in a text file with the extension .out in the same directory as the data file.

The Appendix shows the input and output files `Example_1.txt` and `Example_1.out`

Example 2: Line of Best Fit – correlated data of varying precision



Point	x (mm)	y (mm)
1	–40.0	–24.0
2	–15.0	–24.0
3	10.0	–12.0
4	38.0	15.0
5	67.0	30.0

Figure 2. Line of Best Fit through data points 1 to 5 that have varying precision

The line of best fit shown in the Figure 2 has the equation $y = bx + c$ where b is the slope of the line $\left(b = \tan\theta = \dfrac{y_2 - y_1}{x_2 - x_1} \right)$ and c is the intercept of the line on the y axis.

b and c are the <u>parameters</u> and the data points are assumed to accord with the <u>mathematical model</u> $y = bx + c$ and the x,y coordinate pairs of each data point are considered as <u>indirect</u> measurements of the parameters b and c of the mathematical model.

The data points in Figure 2 have varying precision indicated by error ellipses and the size, shape and orientation of the error ellipses are functions of the variances and covariance of the coordinates at each point. As a general rule, a point with a small error ellipse is more precisely located than a point with a large error ellipse.

The cofactor matrix **Q** of the n = 10 measurements (the coordinates) has the following form

$$\mathbf{Q}_{(n,n)} = \left[\begin{array}{cc:cc:ccc} s_{x_1}^2 & s_{x_1 y_1} & 0 & 0 & 0 & \cdots & 0 \\ s_{x_1 y_1} & s_{y_1}^2 & 0 & 0 & 0 & \cdots & 0 \\ \hdashline 0 & 0 & s_{x_2}^2 & s_{x_2 y_2} & 0 & \cdots & 0 \\ 0 & 0 & s_{x_2 y_2} & s_{y_2}^2 & 0 & \cdots & 0 \\ \hdashline 0 & 0 & 0 & 0 & \ddots & & \vdots \\ \vdots & \vdots & \vdots & \vdots & & s_{x_5}^2 & s_{x_5 y_5} \\ 0 & 0 & 0 & 0 & \cdots & s_{x_5 y_5} & s_{y_5}^2 \end{array}\right]$$

Where $s_{x_k}^2, s_{y_k}^2$ are estimates of the variances σ_x^2 and σ_y^2 respectively and $s_{x_k y_k}$ is an estimate of the covariance σ_{xy}. s_x, s_y are estimates of the standard deviations σ_x, σ_y and standard deviation is the positive square root of the variance.

Note that correlation $\rho_{xy} = \dfrac{s_{xy}}{s_x\, s_y}$ and $-1 \le \rho_{xy} \le 1$

The actual numeric values (mm^2) for each point are:

$$
\mathbf{Q}_{(n,n)} = \left[\begin{array}{cc:cc:cc:cc:cc}
2 & 0.5 & & & & & & & & \\
0.5 & 3 & & & & & & & & \\ \hdashline
 & & 8 & -4 & & & & & & \\
 & & -4 & 5 & & & & & & \\ \hdashline
 & & & & 8 & -3 & & & & \\
 & & & & -3 & 7 & & & & \\ \hdashline
 & & & & & & 1 & 0.5 & & \\
 & & & & & & 0.5 & 2 & & \\ \hdashline
 & & & & & & & & 6 & 1 \\
 & & & & & & & & 1 & 12
\end{array}\right]
$$

Assuming that residuals v_x and v_y are associated with both the *x* and *y* measurements (the coordinate pairs) the observation equation is

$$y_k + v_{y_k} = b\left(x_k + v_{x_k}\right) + c \tag{80}$$

Let
$$b = b^0 + \delta b \tag{81}$$

where b^0 is an approximate value and δb is a small correction and substituting (81) into (80) gives

$$\begin{aligned} y_k + v_{y_k} &= \left(b^0 + \delta b\right)\left(x_k + v_{x_k}\right) + c \\ &= b^0 x_k + b^0 v_{x_k} + x_k \delta b + v_{x_k} \delta b + c \end{aligned} \tag{82}$$

But, since v_{x_k} and δb are both small, then the product $v_{x_k}\delta b \approx 0$ in (82) and we write the observation equation as

$$y_k + v_{y_k} = b^0 x_k + b^0 v_{x_k} + x_k \delta b + c \tag{83}$$

Re-arranging the observation equation so that unknown quantities are on the left-hand-side and known quantities are on the right-hand-side of the equals sign gives

$$-b^0 v_{x_k} + v_{y_k} - x_k \delta b - c = b^0 x_k - y_k \tag{84}$$

For the 5 data pairs ($n = 10$ observations), the ($m = 5$) observation equations for the ($u = 2$) parameters are

$$
\begin{aligned}
-b^0 v_{x_1} + v_{y_1} - x_1\delta b - c &= b^0 x_1 - y_1 \\
-b^0 v_{x_2} + v_{y_2} - x_2\delta b - c &= b^0 x_2 - y_2 \\
-b^0 v_{x_3} + v_{y_3} - x_3\delta b - c &= b^0 x_3 - y_3 \\
-b^0 v_{x_4} + v_{y_4} - x_4\delta b - c &= b^0 x_4 - y_4 \\
-b^0 v_{x_5} + v_{y_5} - x_5\delta b - c &= b^0 x_5 - y_5
\end{aligned}
$$

These can be written in the matrix form $\mathbf{A}_{(m,n)}\mathbf{v}_{(n,1)} + \mathbf{B}_{(m,u)}\delta\mathbf{x}_{(u,1)} = \mathbf{f}_{(m,1)}$ (Combined Case: $\mathbf{A}$; $\mathbf{B}$; $\mathbf{W}; \mathbf{W}_{xx} = \mathbf{0}$) as

$$
\begin{bmatrix}
-b^0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -b^0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -b^0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -b^0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -b^0 & 1
\end{bmatrix}
\begin{bmatrix}
v_{x1} \\ v_{y1} \\ v_{x2} \\ v_{y2} \\ v_{x3} \\ v_{y3} \\ v_{x4} \\ v_{y4} \\ v_{x5} \\ v_{y5}
\end{bmatrix}
+
\begin{bmatrix}
-x_1 & -1 \\ -x_2 & -1 \\ -x_3 & -1 \\ -x_4 & -1 \\ -x_5 & -1
\end{bmatrix}
\begin{bmatrix}
\delta b \\ c
\end{bmatrix}
=
\begin{bmatrix}
b^0 x_1 - y_1 \\ b^0 x_2 - y_2 \\ b^0 x_3 - y_3 \\ b^0 x_4 - y_4 \\ b^0 x_5 - y_5
\end{bmatrix}
$$

The least squares solution for the parameters δb and c and the related precision estimation are given by equations (47) to (57). The solution is iterative, terminating when δb reaches a sufficiently small value.

For a first iteration with an approximate value $b^0 = 0.55$ the numerical values in the matrices **A**, **B** and **f** are

$$
\mathbf{A}_{(m,n)} =
\begin{bmatrix}
-0.55 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -0.55 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -0.55 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -0.55 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -0.55 & 1
\end{bmatrix}
$$

$$
\mathbf{B}_{(m,u)} =
\begin{bmatrix}
40 & -1 \\ 15 & -1 \\ -10 & -1 \\ -38 & -1 \\ -67 & -1
\end{bmatrix}
\quad
\mathbf{f}_{(m,1)} =
\begin{bmatrix}
2.00 \\ 15.75 \\ 17.50 \\ 5.90 \\ 6.85
\end{bmatrix}
$$

and the equivalent weight matrix $\mathbf{W}_{e(m,m)} = \left(\mathbf{A}_{(m,n)}\mathbf{Q}_{(n,n)}\mathbf{A}^T_{(n,m)}\right)^{-1}$ is

$$\mathbf{W}_{e\,(m,m)} = \begin{bmatrix} 0.327332242 & 0 & 0 & 0 & 0 \\ 0 & 0.084602369 & 0 & 0 & 0 \\ 0 & 0 & 0.078616352 & 0 & 0 \\ 0 & 0 & 0 & 0.570613409 & 0 \\ 0 & 0 & 0 & 0 & 0.078647267 \end{bmatrix}$$

The matrices **N** and **t** are

$$\mathbf{N}_{(u,u)} = \mathbf{B}^T_{(u,m)}\mathbf{W}_{e\,(m,m)}\mathbf{B}_{(m,u)} = \begin{bmatrix} 1727.642100568 & 13.376514747 \\ 13.376514747 & 1.139811640 \end{bmatrix}$$

$$\mathbf{t}_{(u,1)} = \mathbf{B}^T_{(u,m)}\mathbf{W}_{e\,(m,m)}\mathbf{f}_{(m,1)} = \begin{bmatrix} -131.610662196 \\ -7.268290852 \end{bmatrix}$$

The solutions are

$$\delta\mathbf{x}_{(u,1)} = \mathbf{N}^{-1}_{(u,u)}\mathbf{t}_{(u,1)} = \begin{bmatrix} \delta b \\ c \end{bmatrix} = \begin{bmatrix} -0.029485717 \\ -6.030711114 \end{bmatrix} \quad \text{and } b = b^0 + \delta b = 0.520514283$$

A <u>second iteration</u> with $b^0 = 0.520514283$ gives the solutions as

$$\delta\mathbf{x}_{(u,1)} = \mathbf{N}^{-1}_{(u,u)}\mathbf{t}_{(u,1)} = \begin{bmatrix} \delta b \\ c \end{bmatrix} = \begin{bmatrix} 0.000359159 \\ -6.083116111 \end{bmatrix} \quad \text{and } b = b^0 + \delta b = 0.520873442$$

A <u>third iteration</u> with $b^0 = 0.520873442$ gives the solutions as

$$\delta\mathbf{x}_{(u,1)} = \mathbf{N}^{-1}_{(u,u)}\mathbf{t}_{(u,1)} = \begin{bmatrix} \delta b \\ c \end{bmatrix} = \begin{bmatrix} -0.000004552 \\ -6.082457028 \end{bmatrix} \quad \text{and } b = b^0 + \delta b = 0.520868890$$

A <u>fourth (and last) iteration</u> with $b^0 = 0.520868890$ gives the solutions as

$$\delta\mathbf{x}_{(u,1)} = \mathbf{N}^{-1}_{(u,u)}\mathbf{t}_{(u,1)} = \begin{bmatrix} \delta b \\ c \end{bmatrix} = \begin{bmatrix} 0.000000058 \\ -6.082465379 \end{bmatrix} \quad \text{and } b = b^0 + \delta b = 0.520868948$$

and the residuals

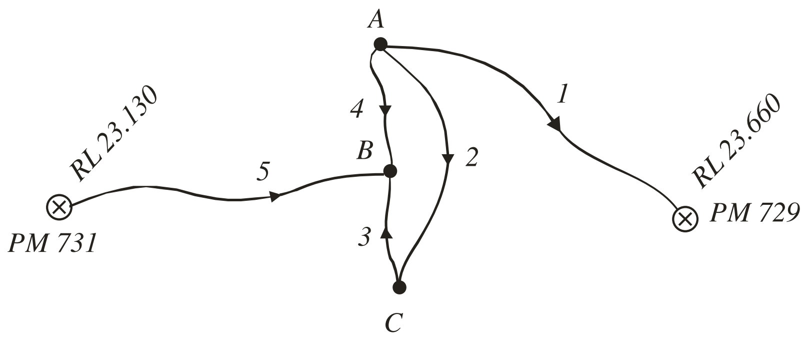
$$\mathbf{v}_{(n,1)} = \mathbf{Q}_{(n,n)}\mathbf{A}^T_{(m,n)}\mathbf{k}_{(n,1)} = \begin{bmatrix} v_{x1} \\ v_{y1} \\ v_{x2} \\ v_{y2} \\ v_{x3} \\ v_{y3} \\ v_{x4} \\ v_{y4} \\ v_{x5} \\ v_{y5} \end{bmatrix} = \begin{bmatrix} 0.523 \\ -2.645 \\ -7.279 \\ 6.313 \\ -6.485 \\ 7.748 \\ 0.015 \\ -1.281 \\ 0.200 \\ -1.080 \end{bmatrix}$$

A MATLAB program *linear_regression_CLS.m* is given in the Appendix that solves Linear Regression problems using Combined Least Squares. This program requires a data file (an ASCII text file) with an extension .txt containing data for each point (and there should be at least three points). each line of the data file contains a point number, x-coordinate, s_x (standard deviation of the x-coordinate), y-coordinate, s_y (standard deviation of the y-coordinate) and s_{xy} (the covariance between the x and y coordinates).

The output from the program containing estimates of parameters and residuals and relevant cofactor matrices is placed in a text file with the extension .out in the same directory as the data file.

The Appendix shows the input and output files **`Example_2.txt`** and **`Example_2.out`**

Example 3: Level Network Adjustment



Line	Height diff	s.d.
1	1.450	0.005
2	0.405	0.002
3	0.655	0.002
4	1.070	0.002
5	0.145	0.005

Figure 3. Level Network

Figure 3 is a schematic diagram of a small level network connecting points *A, B* and *C* to PM's 729 and 731 of known Australian Height datum (AHD) Reduced Levels (RL's). On the diagram, the arrows indicate the direction of rise; i.e., *A* is lower than PM729 and *C*. And *B* is higher than *A, C* and PM731. The height differences and standard deviations (metres) are shown in the table to the right of the diagram.

We will adjust this level network using two different methods; firstly using Observations Only least squares (Condition case) and secondly using Indirect least squares (Parametric case).

Observations Only Least Squares

There are n =5 observations (measured height differences) and a minimum of $n_0 = 3$ observations are required to fix the RL's of *A, B* and *C*. Hence there are $m = n - n_0 = 2$ redundant measurements, which equals the number of independent condition equations. Denoting the observations as $l_1, l_2, \ldots$ etc. these two conditions are

$$\begin{aligned} l_5 + v_5 - (l_4 + v_4) + (l_1 + v_1) &= \mathrm{RL}_{729} - \mathrm{RL}_{731} \\ l_2 + v_2 + l_3 + v_3 - (l_4 + v_4) &= 0 \end{aligned} \tag{85}$$

These equations may be re-arranged with the residuals on the left-hand side of the equals sign and numeric values on the right-hand side

$$\begin{aligned} v_1 - v_4 + v_5 &= 0.530 - (l_1 - l_4 + l_5) \\ v_2 + v_3 - v_4 &= 0 - (l_2 + l_3 - l_4) \end{aligned}$$

The condition equations in matrix form $\mathbf{A}_{(m,n)}\mathbf{v}_{(n,1)} = \mathbf{f}_{(m,1)}$ are

$$\begin{bmatrix} 1 & 0 & 0 & -1 & 1 \\ 0 & 1 & 1 & -1 & 0 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{bmatrix} = \begin{bmatrix} 0.005 \\ 0.010 \end{bmatrix} \tag{86}$$

The cofactor matrix **Q** containing estimates of variances and covariances is (upper-triangular part)

$$\mathbf{Q}_{(n,n)} = \begin{bmatrix} (0.005)^2 & 0 & 0 & 0 & 0 \\ & (0.002)^2 & 0 & 0 & 0 \\ & & (0.002)^2 & 0 & 0 \\ & & & (0.002)^2 & 0 \\ & & & & (0.005)^2 \end{bmatrix} \tag{87}$$

The solution for the residuals **v**, adjusted observations $\hat{\mathbf{l}} = \mathbf{l} + \mathbf{v}$ and cofactor matrices $\mathbf{Q}_{\hat{l}\hat{l}}$ and $\mathbf{Q}_{vv}$ are given by equations (68) to (75). Using (86) and (87) the relevant equations are

$$\begin{aligned} \mathbf{W}_{e\,(m,m)} &= \left(\mathbf{A}_{(m,n)}\mathbf{Q}_{(n,n)}\mathbf{A}^T_{(n,m)}\right)^{-1} = \begin{bmatrix} 18987.341772 & -6329.113924 \\ -6329.113924 & 85443.037975 \end{bmatrix} \\ \mathbf{k}_{(m,1)} &= \mathbf{W}_{e\,(m,m)}\mathbf{f}_{(m,1)} = \begin{bmatrix} 31.645570 \\ 822.784810 \end{bmatrix} \end{aligned} \tag{88}$$

$$\mathbf{v}_{(n,1)} = \mathbf{Q}_{(n,n)}\mathbf{A}^T_{(n,m)}\mathbf{k}_{(m,1)} = \begin{bmatrix} 0.000791 \\ 0.003291 \\ 0.003291 \\ -0.003418 \\ 0.000791 \end{bmatrix} \quad \hat{\mathbf{l}}_{(n,1)} = \mathbf{l}_{(n,1)} + \mathbf{v}_{(n,1)} = \begin{bmatrix} 1.451 \\ 0.408 \\ 0.658 \\ 1.067 \\ 0.146 \end{bmatrix} \tag{89}$$

The adjusted RL's are: A = 22.209 m, B = 23.276 m, C = 22.617 m.

The cofactor matrices of the adjusted height differences and residuals are (upper-triangular part)

$$\mathbf{Q}_{\hat{l}\hat{l}} = \begin{bmatrix} 1.3133\text{e-}05 & 6.3291\text{e-}07 & 6.3291\text{e-}07 & 1.2658\text{e-}06 & -1.1867\text{e-}05 \\ & 2.6329\text{e-}06 & -1.3671\text{e-}06 & 1.2658\text{e-}06 & 6.3291\text{e-}07 \\ & & 2.6329\text{e-}06 & 1.2658\text{e-}06 & 6.3291\text{e-}07 \\ & & & 2.5316\text{e-}06 & 1.2658\text{e-}06 \\ & & & & 1.3133\text{e-}05 \end{bmatrix} \tag{90}$$

$$\mathbf{Q}_{vv} = \begin{bmatrix} 1.1867\text{e-05} & -6.3291\text{e-07} & -6.3291\text{e-07} & -1.2658\text{e-06} & 1.1867\text{e-05} \\ & 1.3671\text{e-06} & 1.3671\text{e-06} & -1.2658\text{e-06} & -6.3291\text{e-07} \\ & & 1.3671\text{e-06} & -1.2658\text{e-06} & -6.3291\text{e-07} \\ & & & 1.4684\text{e-06} & -1.2658\text{e-06} \\ & & & & 1.1867\text{e-05} \end{bmatrix} \tag{91}$$

The variance factor is $$\sigma_0^2 = \frac{\mathbf{v}^T\mathbf{W}\mathbf{v}}{m} = 4.193038 \text{ m}^2 \tag{92}$$

Precision estimates of the adjusted RL's of *A, B* and *C* can be made in the following way.
The RL's are obtained from adjusted observations as

$$\begin{aligned} \text{RL}_A &= \hat{l}_5 - \hat{l}_4 + \text{RL}_{731} \\ \text{RL}_B &= \hat{l}_5 + \text{RL}_{731} \\ \text{RL}_C &= \hat{l}_5 - \hat{l}_3 + \text{RL}_{731} \end{aligned} \quad \text{or} \quad \begin{bmatrix} \text{RL}_A \\ \text{RL}_B \\ \text{RL}_C \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{l}_1 \\ \hat{l}_2 \\ \hat{l}_3 \\ \hat{l}_4 \\ \hat{l}_5 \end{bmatrix} + \begin{bmatrix} \text{RL}_{731} \\ \text{RL}_{731} \\ \text{RL}_{731} \end{bmatrix} \tag{93}$$

These equations can be written in a matrix form $\mathbf{y} = \mathbf{C}\hat{\mathbf{l}} + \mathbf{d}$ and using *Propagation of Variances* for linear functions we may write the cofactor matrix of the parameters in **y** (the adjusted RL's) as

$$\mathbf{Q}_{yy} = \mathbf{C}\mathbf{Q}_{\hat{l}\hat{l}}\mathbf{C}^T \tag{94}$$

where $\mathbf{Q}_{yy}$ contains estimates of the variances $\left(s_A^2, s_B^2, s_C^2\right)$ and covariances $\left(s_{AB}, s_{AC}, s_{BC}\right)$ of the adjusted RL's.

$$\mathbf{Q}_{yy} = \begin{bmatrix} s_A^2 & s_{AB} & s_{AC} \\ s_{BA} & s_B^2 & s_{BC} \\ s_{CA} & s_{CB} & s_C^2 \end{bmatrix}$$

with **C** given in (93) and $\mathbf{Q}_{\hat{l}\hat{l}}$ given in (90) the cofactor matrix of the adjusted RL's $\mathbf{Q}_{yy}$ in (94) is

$$\mathbf{Q}_{yy} = \begin{bmatrix} 1.313291\text{e-05} & 1.186709\text{e-05} & 1.250000\text{e-05} \\ 1.186709\text{e-05} & 1.313291\text{e-05} & 1.250000\text{e-05} \\ 1.250000\text{e-05} & 1.250000\text{e-05} & 1.450000\text{e-05} \end{bmatrix} \tag{95}$$

Using the relationship $\mathbf{\Sigma} = \sigma_0^2\mathbf{Q}$ and (95) the standard deviations of the adjusted RL's are

$$\begin{aligned} \sigma_A &= \sqrt{\sigma_0^2\, s_A^2} = \sqrt{(4.193038)(1.313291\text{e-05})} = 0.007421 \text{ m} \\ \sigma_B &= \sqrt{\sigma_0^2\, s_B^2} = \sqrt{(4.193038)(1.313291\text{e-05})} = 0.007421 \text{ m} \\ \sigma_C &= \sqrt{\sigma_0^2\, s_C^2} = \sqrt{(4.193038)(1.450000\text{e-05})} = 0.007797 \text{ m} \end{aligned}$$

Indirect Least Squares

The observation equation for a measured height difference l_{XY} between two points *X* and *Y* can be written as

$$RL_X + l_{XY} + v_{XY} = RL_Y \tag{96}$$

Using (96) we may write an equation for each of the *n* = 5 observations (measured height differences) in the form

$$\begin{aligned}
\mathrm{RL}_A + l_1 + v_1 &= \mathrm{RL}_{729} \\
\mathrm{RL}_A + l_2 + v_2 &= \mathrm{RL}_C \\
\mathrm{RL}_C + l_3 + v_3 &= \mathrm{RL}_B \\
\mathrm{RL}_A + l_4 + v_4 &= \mathrm{RL}_B \\
\mathrm{RL}_{731} + l_5 + v_5 &= \mathrm{RL}_B
\end{aligned}$$

These equations may be re-arranged so that the unknown quantities (the residuals and the *u* = 3 unknown RL's of *A,B,C*) are on the left-hand side of the equals sign and the known quantities are on the right-hand side

$$\begin{array}{llll}
v_1 + \mathrm{RL}_A & + \mathrm{RL}_B & + \mathrm{RL}_C & = \mathrm{RL}_{729} - l_1 \\
v_2 + \mathrm{RL}_A & & + \mathrm{RL}_C & = -l_2 \\
v_3 & - \mathrm{RL}_B & + \mathrm{RL}_C & = -l_3 \\
v_4 + \mathrm{RL}_A & - \mathrm{RL}_B & & = -l_4 \\
v_5 & - \mathrm{RL}_B & & = -\left(\mathrm{RL}_{731} + l_5\right)
\end{array}$$

In the matrix form $\mathbf{v}_{(n,1)} + \mathbf{B}_{(n,u)}\mathbf{x}_{(u,1)} = \mathbf{f}_{(n,1)}$ the equations are

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & -1 \\ 0 & -1 & 1 \\ 1 & -1 & 0 \\ 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} \mathrm{RL}_A \\ \mathrm{RL}_B \\ \mathrm{RL}_C \end{bmatrix} = \begin{bmatrix} 22.210 \\ -0.405 \\ -0.655 \\ -1.070 \\ -23.275 \end{bmatrix} \tag{97}$$

The estimates of the variances and covariances are contained in **Q** [see (87)] and the diagonal weight matrix **W** is

$$\mathbf{W}_{(n,n)} = \mathbf{Q}^{-1}_{(n,n)} = \begin{bmatrix} 40000 & & & & \\ & 250000 & & & \\ & & 250000 & & \\ & & & 250000 & \\ & & & & 40000 \end{bmatrix} \tag{98}$$

The solution for the vector **x** (the three RL's), the residuals **v** and cofactor matrices $\mathbf{Q}_{xx}$ and $\mathbf{Q}_{vv}$ are given by equations (58) to (67) and these are embodied in the MATLAB function *least_squares.m* given in the Appendix, and a suitable ASCII text file for this problem is

```
% Data file for function "least_squares.m"
% Example 3: Level Network
%  B(1)  B(2)  B(3)   f        w
    1     0     0   22.210   40000
    1     0    -1   -0.405  250000
    0    -1     1   -0.655  250000
    1    -1     0   -1.070  250000
    0    -1     0  -23.275   40000
```

The solutions are

$$\mathbf{N}_{(u,u)} = \mathbf{B}^T_{(u,n)}\mathbf{W}_{(n,n)}\mathbf{B}_{(n,u)} = \begin{bmatrix} 540000 & -250000 & -250000 \\ -250000 & 540000 & -250000 \\ -250000 & -250000 & 500000 \end{bmatrix}$$

$$\mathbf{t}_{(u,1)} = \mathbf{B}^T_{(u,n)}\mathbf{W}_{(n,n)}\mathbf{f}_{(n,1)} = \begin{bmatrix} 519650 \\ 1362250 \\ -62500 \end{bmatrix}$$

$$\mathbf{x}_{(u,1)} = \mathbf{N}^{-1}_{(u,u)}\mathbf{t}_{(u,1)} = \begin{bmatrix} 1.3133\text{e-}05 & 1.1867\text{e-}05 & 1.2500\text{e-}05 \\ 1.1867\text{e-}05 & 1.3133\text{e-}05 & 1.2500\text{e-}05 \\ 1.2500\text{e-}05 & 1.2500\text{e-}05 & 1.4500\text{e-}05 \end{bmatrix} \begin{bmatrix} 519650 \\ 1362250 \\ -62500 \end{bmatrix} = \begin{bmatrix} 22.209209 \\ 23.275791 \\ 22.617500 \end{bmatrix}$$

$$\mathbf{v}_{(n,1)} = \mathbf{f}_{(n,1)} - \mathbf{B}_{(n,u)}\mathbf{x}_{(u,1)} = \begin{bmatrix} 0.000791 \\ 0.003291 \\ 0.003291 \\ -0.003418 \\ 0.000791 \end{bmatrix}$$

The cofactor matrix $\mathbf{Q}_{xx} = \begin{bmatrix} s_A^2 & s_{AB} & s_{AC} \\ s_{BA} & s_B^2 & s_{BC} \\ s_{CA} & s_{CB} & s_C^2 \end{bmatrix} = \mathbf{N}^{-1} = \begin{bmatrix} 1.3133\text{e-}05 & 1.1867\text{e-}05 & 1.2500\text{e-}05 \\ 1.1867\text{e-}05 & 1.3133\text{e-}05 & 1.2500\text{e-}05 \\ 1.2500\text{e-}05 & 1.2500\text{e-}05 & 1.4500\text{e-}05 \end{bmatrix}$

and the cofactor matrices $\mathbf{Q}_{\hat{l}\hat{l}}$ and $\mathbf{Q}_{vv}$ are given in (90) and (91).

The variance factor is $\sigma_0^2 = \dfrac{\mathbf{v}^T\mathbf{W}\mathbf{v}}{n-u} = 4.193038 \text{ m}^2$

Using the relationship $\mathbf{\Sigma}_{xx} = \sigma_0^2\mathbf{Q}_{xx}$ the standard deviations of the adjusted RL's of *A*, B and *C* are

$$\sigma_A = \sqrt{\sigma_0^2\, s_A^2} = \sqrt{(4.193038)(1.3133\text{e-}05)} = 0.007421 \text{ m}$$
$$\sigma_B = \sqrt{\sigma_0^2\, s_B^2} = \sqrt{(4.193038)(1.3133\text{e-}05)} = 0.007421 \text{ m}$$
$$\sigma_C = \sqrt{\sigma_0^2\, s_C^2} = \sqrt{(4.193038)(1.4500\text{e-}05)} = 0.007797 \text{ m}$$

Example 4: Position Fix by measured distances

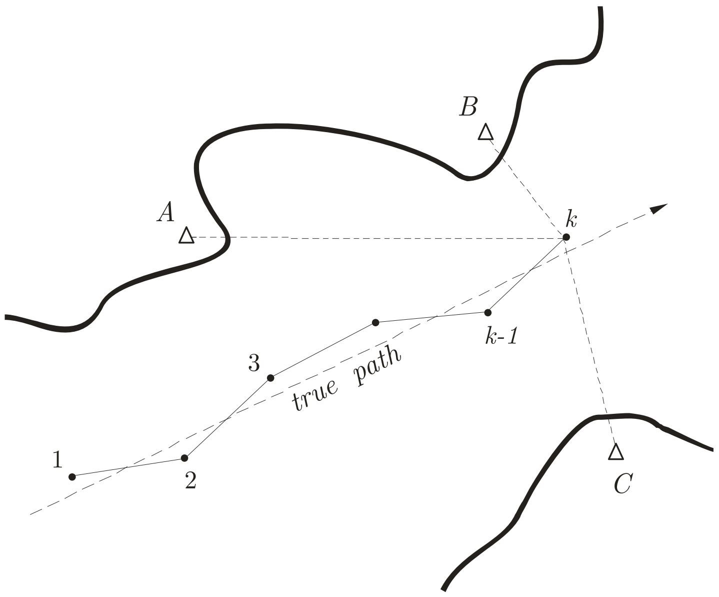


Figure 4. Path of a ship in a navigation channel

Figure 4 shows the path of a ship in a navigation channel as it moves down the shipping channel at a constant heading and speed. Navigation equipment on board automatically measures distances to transponders at three navigation beacons *A, B* and *C* at 60-second intervals. The measured distances are known to have a standard deviation of 1 metre and the solid line in Figure 4 represents solutions of the ship's position for each set of measurements at the 60-second time intervals. The true path of the ship is shown as the dotted line.

The coordinates of the three navigation beacons are:

A:	10000.000 E	*B:*	13880.000 E	*C:*	15550.000 E
	10000.000 N		11250.000 N		7160.000 N

When the ship was at position 1 the measurements to the beacons were:

$1 \rightarrow A$: 4249.7 m $1 \rightarrow B$: 7768.6 m $1 \rightarrow C$: 7721.1 m

And the approximate location of the ship at position 1 is: 7875.000 E, 6319.392 N.

Indirect Least Squares can be used to determine the best estimate of the ship at position 1 by an iterative technique set out as follows.

Observation equation for a measured distance

The observation equation for a measured distance at position *k* to beacon *j* can be written as

$$l_{kj} + v_{kj} = \hat{l}_{kj} \tag{99}$$

where l_{kj} are the measured distance, v_{kj} is the residual (small unknown correction) and $\hat{l}_{kj}$ is the least squares estimate. $k = 1,2,3, \ldots$ is the ship location and $j = A,B,C$ are the beacons.

The estimates $\hat{l}_{kj}$ are non-linear functions of the beacon coordinates E_j, N_j and the ship estimates $\hat{E}_k, \hat{N}_k$

$$\hat{l}_{kj} = \hat{l}\left(\hat{E}_k, \hat{N}_k, E_j, N_j\right) = \sqrt{\left(\hat{E}_k - E_j\right)^2 + \left(\hat{N}_k - N_j\right)^2} \tag{100}$$

Expanding (100) into a series using Taylor's theorem gives

$$\hat{l} = l' + \frac{\partial \hat{l}}{\partial \hat{E}_k}\left(\hat{E}_k - E'_k\right) + \frac{\partial \hat{l}}{\partial \hat{N}_k}\left(\hat{N}_k - N'_k\right) + \text{ higher-order terms} \tag{101}$$

where E'_k, N'_k are approximate coordinates of the ship at position *k*, l' is an approximate distance computed using E'_k, N'_k and the coordinates of the beacon, and the partial derivatives are

$$\frac{\partial \hat{l}}{\partial \hat{E}_k} = \frac{E'_k - E_j}{l'_{kj}} = d_{kj} \; ; \quad \frac{\partial \hat{l}}{\partial \hat{N}_k} = \frac{N'_k - N_j}{l'_{kj}} = c_{kj} \quad \text{for} \quad j = A, B, C \tag{102}$$

d_{kj}, c_{kj} are dimensionless quantities known as distance coefficients.

With $\hat{E} = E' + \delta E$ and $\hat{N} = N' + \delta N$ where $\delta E, \delta N$ are small corrections then $\hat{E} - E' = \delta E$ and $\hat{N} - N' = \delta N$. These expressions can be substituted into (101) – ignoring higher-order terms – giving a Taylor series approximation for $\hat{l}$ for a single distance

$$\hat{l} = l' + d\,\delta E + c\,\delta N \tag{103}$$

Re-arranging (99) for a single distance gives

$$v - \hat{l} = -l \tag{104}$$

and substituting (103) into (104) and re-arranging gives the linearized form of the observation equation for a measured distance as

$$v_{kj} - d_{kj}\,\delta E_k - c_{kj}\,\delta N_k = l'_{kj} - l_{kj} \tag{105}$$

Matrix form of observation equations

Observation equations for each measured distance to beacons $j = A, B, C$ from location $k = 1$ can be written in matrix form as

$$\begin{bmatrix} v_A \\ v_B \\ v_C \end{bmatrix}_{k=1} + \begin{bmatrix} -d_A & -c_A \\ -d_B & -c_B \\ -d_C & -c_C \end{bmatrix}_{k=1} \begin{bmatrix} \delta E \\ \delta N \end{bmatrix}_{k=1} = \begin{bmatrix} l'_A - l_A \\ l'_B - l_B \\ l'_C - l_C \end{bmatrix}_{k=1} \tag{106}$$

The *n* = 3 equations in *u* = 2 unknowns $(\delta E, \delta N)$ can be written as

$$\mathbf{v}_{(n,1)} + \mathbf{B}_{(n,u)}\delta\mathbf{x}_{(u,1)} = \mathbf{f}_{(n,1)} \tag{107}$$

The Indirect Least Squares solution for the vector of corrections $\delta\mathbf{x}$ is given by equations (58) to (67). The approximate coordinates are 'updated' by adding the corrections and a new iteration (new equations (106) formed and solved) performed. The iterative process is terminated when the corrections reach some pre-determined (small) value.

Iterative solution

1. Coordinates, measured distances and standard deviations

Station	East (m)	North (m)	Observed distance l (m)	s.d. (m)
1	$E' = 7875.000$	$N' = 6319.392$		
A	10000.000	10000.000	4249.7	1.0
B	13880.000	11250.000	7768.6	1.0
C	15550.000	7160.000	7721.1	1.0

2. Computed bearings and distances, distance coefficients, numeric terms (comp − obs)

Station	Bearing (degrees)	Distance l' (m)	$d = \frac{E'-E}{l'}$	$c = \frac{N'-N}{l'}$	$l' - l$
1					
A	30.000000	4250.000029	-0.500000	-0.866025	0.300029
B	50.611138	7769.872602	-0.772857	-0.634580	1.272602
C	83.749566	7720.896762	-0.994056	-0.108874	-0.203238

3. Solution for small corrections $\delta E, \delta N$

The solution for the vector $\delta\mathbf{x}$ (the corrections $\delta E, \delta N$), the residuals $\mathbf{v}$ and cofactor matrices $\mathbf{Q}_{xx}$ and $\mathbf{Q}_{vv}$ are given by equations (58) to (67) and these are embodied in the MATLAB function *least_squares.m* given in the Appendix, and a suitable ASCII text file for this problem is

```
% Data file for function "least_squares.m"
% Example 4: Position Fix by distances
%  B(1)      B(2)       f        w
   0.500000  0.866025  0.300029  1
   0.772857  0.634580  1.272602  1
   0.994056  0.108874 -0.203238  1
```

With $\mathbf{W} = \mathbf{I}$ the solutions are

$$\mathbf{N}_{(u,u)} = \mathbf{B}^T_{(u,n)}\mathbf{W}_{(n,n)}\mathbf{B}_{(n,u)} = \begin{bmatrix} 1.835454 & 1.031680 \\ 1.031680 & 1.164546 \end{bmatrix}$$

$$\mathbf{t}_{(u,1)} = \mathbf{B}^T_{(u,n)}\mathbf{W}_{(n,n)}\mathbf{f}_{(n,1)} = \begin{bmatrix} 0.931524 \\ 1.045274 \end{bmatrix}$$

$$\delta\mathbf{x}_{(u,1)} = \mathbf{N}^{-1}_{(u,u)}\mathbf{t}_{(u,1)} = \begin{bmatrix} 1.085209 & -0.961395 \\ -0.961395 & 1.710410 \end{bmatrix}\begin{bmatrix} 0.931524 \\ 1.045274 \end{bmatrix} = \begin{bmatrix} 0.005978 \\ 0.892285 \end{bmatrix}\begin{matrix} \delta E \\ \delta N \end{matrix}$$

$$\mathbf{v}_{(n,1)} = \mathbf{f}_{(n,1)} - \mathbf{B}_{(n,u)}\mathbf{x}_{(u,1)} = \begin{bmatrix} -0.475701 \\ 0.701756 \\ -0.306327 \end{bmatrix} \qquad \hat{\mathbf{l}}_{(n,1)} = \mathbf{l}_{(n,1)} + \mathbf{v}_{(n,1)} = \begin{bmatrix} 4249.224299 \\ 7769.301756 \\ 7720.793673 \end{bmatrix}$$

The cofactor matrix $\mathbf{Q}_{xx} = \mathbf{N}^{-1}$ (see above) and the cofactor matrices $\mathbf{Q}_{\hat{l}\hat{l}}$ and $\mathbf{Q}_{vv}$ are

$$\mathbf{Q}_{vv} = \begin{bmatrix} 2.7848\text{e-}01 & -4.1082\text{e-}01 & 1.7933\text{e-}01 \\ & 6.0604\text{e-}01 & -2.6455\text{e-}01 \\ & & 1.1548\text{e-}01 \end{bmatrix}$$

$$\mathbf{Q}_{\hat{l}\hat{l}} = \begin{bmatrix} 7.2152\text{e-}01 & 4.1082\text{e-}01 & -1.7933\text{e-}01 \\ & 3.9396\text{e-}01 & 2.6455\text{e-}01 \\ & & 8.8452\text{e-}01 \end{bmatrix}$$

The variance factor $\sigma_2^0 = \dfrac{\mathbf{v}^T\mathbf{W}\mathbf{v}}{n-u} = 0.812589 \text{ m}^2$

4. Update coordinates

$$\begin{aligned} E &= E' + \delta E &&= 7875.000 + 0.006 = 7875.006 \text{ m} \\ N &= N' + \delta N &&= 6319.392 + 0.892 = 6320.284 \text{ m} \end{aligned}$$

5. Perform next iteration

   It can be shown that using the updated coordinates in a new iteration yields corrections having magnitudes less than 0.0005 m. So the values E = 7875.006 m, N = 6320.284 m can be regarded as correct to the nearest mm.

Precision of Position Fix

The variance-covariance matrix

$$\mathbf{\Sigma}_{xx} = \begin{bmatrix} \sigma_E^2 & \sigma_{EN} \\ \sigma_{EN} & \sigma_N^2 \end{bmatrix} = \sigma_0^2\mathbf{Q}_{xx} = \sigma_0^2\mathbf{N}^{-1} = \begin{bmatrix} 0.881829 & -0.781219 \\ -0.781219 & 1.389860 \end{bmatrix} \quad (108)$$

contains the variances and covariances of the adjusted coordinates of point 1.

The standard deviations of the east and north coordinates are

$$\begin{aligned} \sigma_E &= \sqrt{0.881829} = 0.939 \text{ m} \\ \sigma_N &= \sqrt{1.389860} = 1.179 \text{ m} \end{aligned}$$

Standard Error Ellipse

Consider a point whose variances σ_E^2, σ_N^2 and covariance σ_{EN} are known. The variance in any other direction *u* may be calculated by considering the projection of *E* and *N* onto the *u*-axis which is rotated anti-clockwise from the *E*-axis by an angle ϕ.

$$u = E\cos\phi + N\sin\phi$$

$$= \begin{bmatrix} \cos\phi & 0 \\ 0 & \sin\phi \end{bmatrix} \begin{bmatrix} E \\ N \end{bmatrix} \qquad (109)$$

or $$\mathbf{y} = \mathbf{A}\mathbf{x} \qquad (110)$$

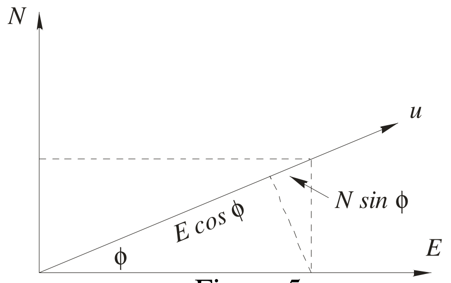


Figure 5.

Applying *Propagation of Variances* to (110) gives $\mathbf{\Sigma}_{yy} = \mathbf{A}\mathbf{\Sigma}_{xx}\mathbf{A}^T$ or

$$\begin{bmatrix} \sigma_u^2 \end{bmatrix} = \begin{bmatrix} \cos\phi & 0 \\ 0 & \sin\phi \end{bmatrix} \begin{bmatrix} \sigma_E^2 & \sigma_{EN} \\ \sigma_{EN} & \sigma_N^2 \end{bmatrix} \begin{bmatrix} \cos\phi & 0 \\ 0 & \sin\phi \end{bmatrix} = \sigma_E^2\cos^2\phi + \sigma_N^2\sin^2\phi + 2\sigma_{EN}\cos\phi\sin\phi \qquad (111)$$

Equation (111) gives the variance σ_u^2 in a direction ϕ (positive anti-clockwise) from the *E*-axis and defines the *pedal curve* of the *Standard Error Ellipse.*

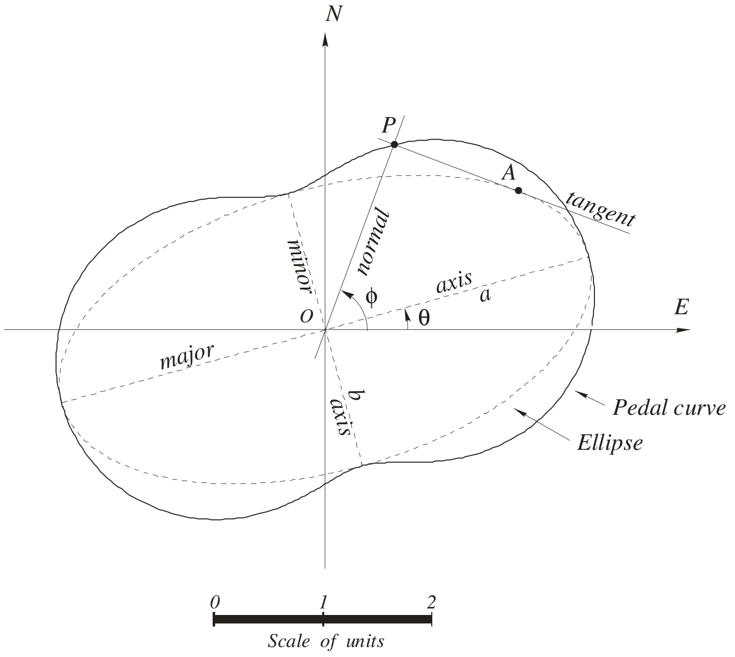


Figure 6. The pedal curve of the Standard Error Ellipse

In Figure 6, *A* is a point on an ellipse. The tangent to the ellipse at *A* intersects a normal to the tangent passing through *O* at *P*. As *A* moves around the ellipse, the locus of all points *P* is the pedal curve of the ellipse. The distance $OP = \sqrt{\sigma_u^2}$ for the angle ϕ. The maximum and minimum values of σ_u^2 define the direction and lengths of the axes of the ellipse.

The semi-axes *a* and *b* of the Standard Error Ellipse are the maximum and minimum values of σ_u and the angle (positive anti-clockwise) from the *E*-axis to the major axis of the Standard Error Ellipse is θ and

$$\begin{aligned} W &= \sqrt{\left(\sigma_E^2 - \sigma_N^2\right)^2 + \left(2\sigma_{EN}\right)^2} \\ a &= \sqrt{\tfrac{1}{2}\left(\sigma_E^2 + \sigma_N^2 + W\right)} \\ b &= \sqrt{\tfrac{1}{2}\left(\sigma_E^2 + \sigma_N^2 - W\right)} \\ \tan 2\theta &= \frac{2\sigma_{EN}}{\sigma_E^2 - \sigma_N^2} \end{aligned} \tag{112}$$

Using the values in (108) $\sigma_E^2 = 0.881829$, $\sigma_N^2 = 1.389860$, $\sigma_{EN} = -0.781219$ and using (112) gives the parameters of the Standard Error Ellipse for the position fix at point 1 as

$$\begin{aligned} W &= 1.642957 \\ a &= 1.399044 \text{ m} \\ b &= 0.560683 \text{ m} \\ \tan 2\theta &= \frac{-1.562438}{-0.508031} \\ 2\theta &= 251.987923 \text{ degree} \\ \theta &= 125.993962 \text{ degree} \end{aligned}$$

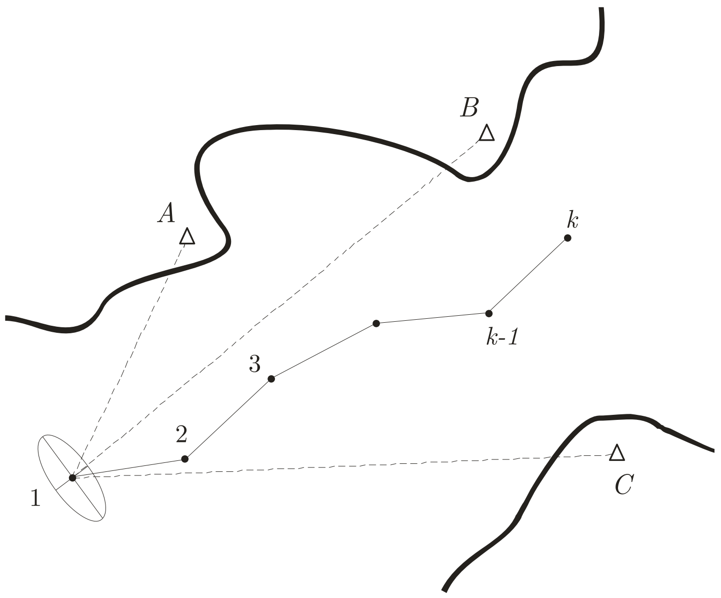


Figure 7. Standard Error Ellipse of position fix 1

Exercise 1: Line of Best Fit

Figure 8 shows part of an Abstract of Fieldnotes with offsets from occupation to a traverse line in Whitten Street. The bearing of the traverse line is 90° 00′.

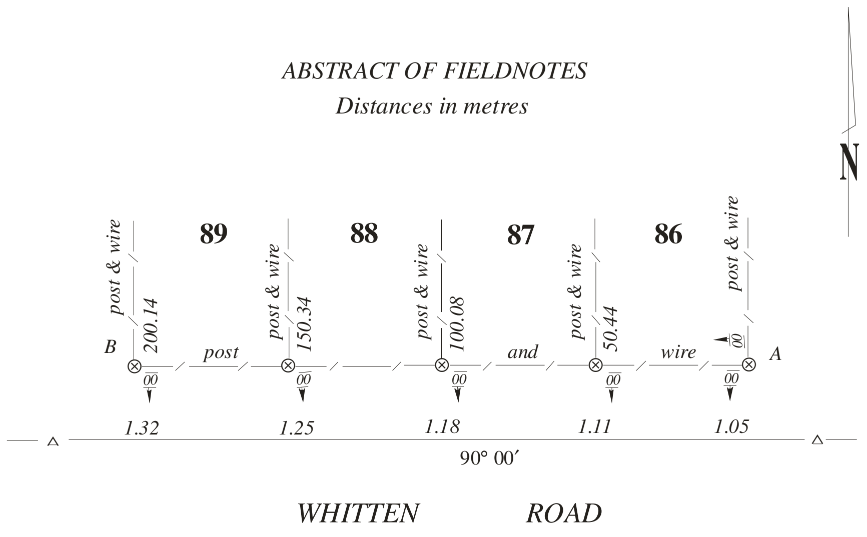


Figure 8. Traverse and offsets in Whitten Road

Use Indirect least squares (and *least_squares.m*) to determine the bearing of the line of best fit through the occupation in Whitten Street. You may consider the chainages (linear distances along Whitten Street) to be free of error.

(The answer is: 90° 04′ 40″)

Exercise 2: Index Error of Total Station EDM

Six horizontal distances between four points in a straight line are measured with a Total Station that is known to have an index correction c.

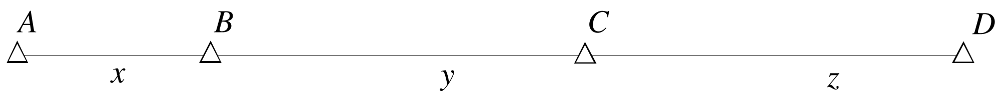


Figure 9. Total Station baseline

The Total Station measurements are:	AB 51.198 m	BC 111.745 m	CD 103.605 m
	AC 162.910 m	BD 215.301 m	
	AD 266.460 m		

The measurements are assumed to be of equal precision (i.e., $\mathbf{W} = \mathbf{I}$).

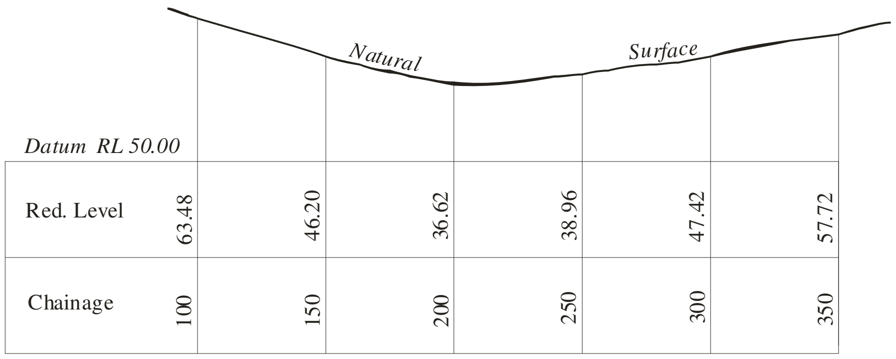
The measurement model is assumed to be: observation + residual + c = true value

Use Indirect least squares (and *least_squares.m*) to determine the index correction c and the distances x, y and z.

(The answers are: c = -0.0440, x = 51.1580 m, y = 111.70215 m, z = 103.5570 m)

Exercise 3: Parabolic Vertical Curve

A surveyor working on the re-alignment of a rural road is required to fit a parabolic vertical curve such that it is the best fit to the series of natural surface Reduced Levels (RLs) on the proposed new alignment. Figure 10 shows a vertical section of the proposed alignment with Chainages (x-values) and RLs (y-values).



Red. Level	63.48	46.20	36.62	38.96	47.42	57.72
Chainage	100	150	200	250	300	350

Figure 10. Natural surface vertical section

The general equation of a parabolic curve is $y = ax^2 + bx + c$

The chainages are assumed to be error free and the RL's are of equal precision.

Use Indirect least squares (and *least_squares.m*) to determine the values of the coefficients a and b and the constant term c.

(The answers are: $a = 0.0015$, $b = -0.6882$, $c = 116.3500$)

Exercise 4: Position Fix

In Example 4 (see Figure 4) the position of the ship in the navigation channel is obtained using Indirect least squares to solve for corrections to approximate coordinates of the ship when the measurements to shore-based beacons are made. In Example 4 the ship is at position 1.

For this Exercise, solve for the ship's position at point 3 when the measured distances are

$3 \rightarrow A$: 3518.4 m $3 \rightarrow B$: 6872.2 m $3 \rightarrow C$: 6857.6 m

And the approximate location of the ship at position 3 is: 8705.5 E, 6727.9 N.

Only a single iteration is required.

(Answer: $E = 8705.803$ m, $N = 6727.959$ m)

THE KALMAN FILTER

A Kalman filter is a set of equations that are applied recursively to estimate the state of a system from a sequence of noisy measurements at times $t_1, t_2, t_3, \dots$ etc. The state of the system is its value or values at times $t_1, t_2, t_3, \dots$ etc. and a system may have a single value or multiple values. Say, for instance, the system is a ship steaming on a particular heading in a shipping channel and the state of the system (the ship) is its east and north coordinates (E_k, N_k) and its velocity $(\dot{E}_k, \dot{N}_k)$ We say that this system (the ship) has a state vector $\mathbf{x}_k = \left[E_k, N_k, \dot{E}_k, \dot{N}_k\right]^T$ containing four elements and the subscript *k* indicates a value at time t_k.

On the other hand, a system may be a process such as electronic distance measurement (EDM) by phase comparison of emitted and reflected light beams. The state of this system is a single value, the distance (D_k), determined at times $t_1, t_2, t_3, \dots$ etc., and this system (the EDM) has a state vector $\mathbf{x}_k = [D_k]$ containing a single element and the subscript *k* indicates a value at time t_k.

Noisy measurements are measurements that contain small random errors assumed to be normally distributed, i.e., the aggregation of errors in size groupings would form the familiar symmetric bell-shaped histogram with positive and negative errors equally likely and small errors more frequent than large errors. Surveyors usually talk of residuals (or corrections) rather than errors, where a residual is the same magnitude as an error but of opposite sign.

A Kalman filter gives the best estimates of the state of a dynamic system at a particular instant of time. And a dynamic system can be one whose values are changing with time, due to the motion of the system and measurement errors, or one whose values are measured at various instants of time and appear to change due to measurement errors. Dynamic systems do not have a single state (consisting of one or many values) that can be determined from a finite set of measurements but instead have a continuously changing state that has values sampled at different instants of time.

The Kalman filter equations were published in 1960 by Dr. R.E. Kalman in his famous paper describing a new approach to the solution of linear filtering and prediction (Kalman 1960). Since that time, papers on the application of the technique have been filling numerous scientific journals and it is regarded as one of the most important algorithmic techniques ever devised. It has been used in applications ranging from navigating the Ranger and Apollo spacecraft in their lunar missions to predicting short-term fluctuations in the stock market. Sorenson (1985) shows Kalman's technique to be an extension of C.F. Gauss' original method of least squares developed in 1795 and provides an historical commentary on its practical solution of linear filtering problems studied by 20th century mathematicians.

The derivation of the Kalman filter equations can be found in many texts related to signal processing that is the usual domain of Electrical Engineers, e.g., Brown and Hwang (1992). These derivations often use terminology that is unfamiliar to surveyors, but two authors, Krakiwsky (1975) and Cross (1992) both with geodesy/surveying backgrounds, have derivations, explanations, terminology and examples that would be familiar to any surveyor. This paper uses terminology similar to Cross and Krakiwsky. The Kalman filter equations and the associated measurement and dynamic models are given below with a brief explanation of the terms. It is hoped that the study of the two examples will make help to make the Kalman filter a relatively easily understood process.

In the derivation and explanation that follows the 'hat' symbol (^) above a vector **x** indicates that it is an estimate of the true (but unknown) state of the system derived from the Kalman Filter (a least squares process). This is also known as the filtered state. The 'prime' symbol ($'$) indicates a predicted quantity.

Equations marked $***$ on the left-hand side are 'components' of the Kalman Filter and are listed in the summary of the filter at the end of the derivations.

Primary and Secondary (or Dynamic) Measurement Models

Suppose that $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \ldots, \mathbf{x}_{k-1}, \mathbf{x}_k$ are vectors of parameters or <u>state vectors</u> of a system at times $t_1, t_2, t_3, \ldots, t_{k-1}, t_k$ and that $\mathbf{l}_1, \mathbf{l}_2, \mathbf{l}_3, \ldots, \mathbf{l}_{k-1}, \mathbf{l}_k$ are the corresponding vectors of measurements associated with the parameters. We may write three equations as follows:

$$\begin{aligned} \mathbf{v}_{k-1} + \mathbf{B}_{k-1}\mathbf{x}_{k-1} &= \mathbf{f}_{k-1} && \text{primary } t_{k-1} \\ \mathbf{v}_k + \mathbf{B}_k\mathbf{x}_k &= \mathbf{f}_k && \text{primary } t_k \\ \mathbf{x}_k &= \mathbf{T}\mathbf{x}_{k-1} + \mathbf{v}_m && \text{secondary or dynamic} \end{aligned} \tag{113}$$

where

x	is the state vector containing the parameters of the system
v	is the vector of residuals associated with the measurements **l** where $\hat{\mathbf{l}} = \mathbf{l} + \mathbf{v}$
B	is a coefficient matrix
f	is a vector of numeric terms derived from the measurements **l**
T	is the <u>transition matrix</u>
$\mathbf{v}_m$	is a vector of residuals associated with the <u>dynamic</u> model

The <u>primary measurement models</u> in (113) link measurements **l** (contained in the vector of numeric terms **f**) with parameters in the state vector **x** at times t_{k-1} and t_k. The primary measurement model is the same as the Parametric case in the earlier Combined Least Squares derivations and examples [see Example 1: *Line of Best Fit* and Example 3: *Level Network Adjustment (Indirect Least Squares)*]

The <u>secondary or dynamic model</u> in (113) links the state vectors **x** at times t_{k-1} and t_k. The transition matrix **T** is an attempt to model temporal changes between the state vectors (the dynamics of the system) and $\mathbf{v}_m$ is a vector of corrections reflecting the fact that the transition matrix **T** is an approximation of the true dynamics. The elements of $\mathbf{v}_m$ are assumed to be small, random and normally distributed with a mean of zero.

The measurements $\mathbf{l}_{k-1}$ and $\mathbf{l}_k$ and the model corrections $\mathbf{v}_m$ have associated weight matrices $\mathbf{W}_{k-1}, \mathbf{W}_k$ and $\mathbf{W}_m$ and cofactor matrices $\mathbf{Q}_{k-1}, \mathbf{Q}_k$ and $\mathbf{Q}_m$ where in general $\mathbf{Q} = \mathbf{W}^{-1}$.

System Driving Noise of the Secondary Model

For the solution of many practical problems, it is useful to assume the vector $\mathbf{v}_m$ as being the product of two matrices

$$\mathbf{v}_m = \mathbf{H}\mathbf{w} \tag{114}$$

where **w** is a vector of quantities known as the system driving noise which cause the secondary model to be incorrect and **H** is a coefficient matrix chosen so that the product **Hw** represents the effect of these quantities on the parameters. Note that in general **H** will not be a square matrix as the number of error sources causing the system noise in the secondary model is not necessarily equal to the number of parameters in **x**.

The Cofactor Matrix of the Secondary Model $\mathbf{Q}_m$

The system driving noise **w** in (114) is assumed to be a vector of random quantities with zero mean and variance matrix estimated by the cofactor matrix $\mathbf{Q}_w$. The cofactor matrix $\mathbf{Q}_m$ can be obtained using *Propagation of Variances* (or propagation of cofactors) that can be summarized as follows: If linear (or linearized) equations can be expressed in a matrix form

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{b}$$

where **y** is a vector of variables, **A** is a coefficient matrix, **x** is a vector of variables having an associated cofactor matrix $\mathbf{Q}_x$ and **b** is a vector of constants then the cofactor matrix of the variables **y** is given by

$$\mathbf{Q}_y = \mathbf{A}\mathbf{Q}_x\mathbf{A}^T$$

Cofactor propagation using (114) gives

$$\mathbf{Q}_m = \mathbf{H}\mathbf{Q}_w\mathbf{H}^T \tag{115}$$

and the weight matrix of the secondary model is given by

$$\mathbf{W}_m = \left(\mathbf{Q}_m\right)^{-1} = \left(\mathbf{H}\mathbf{Q}_w\mathbf{H}^T\right)^{-1} \tag{116}$$

Derivation of the Kalman Filter Equations

The system of equations for the Kalman Filter are the primary models at t_{k-1} and t_k, and the secondary model are

$$\begin{aligned} \mathbf{v}_{k-1} + \mathbf{B}_{k-1}\mathbf{x}_{k-1} &= \mathbf{f}_{k-1} \\ \mathbf{v}_k + \mathbf{B}_k\mathbf{x}_k &= \mathbf{f}_k \\ -\mathbf{T}\mathbf{x}_{k-1} + \mathbf{x}_k - \mathbf{v}_m &= \mathbf{0} \end{aligned} \tag{117}$$

Enforcing the least squares principle – *the sum of the squares of the residuals, multiplied by coefficients reflecting their precision* $\left(\mathbf{v}^T\mathbf{W}\mathbf{v}\right)$ – leads to minimizing the function φ where

$$\begin{aligned}\varphi = \mathbf{v}_{k-1}^T\mathbf{W}_{k-1}\mathbf{v}_{k-1} &+ \mathbf{v}_k^T\mathbf{W}_k\mathbf{v}_k + \mathbf{v}_m^T\mathbf{W}_m\mathbf{v}_m \\ &-2\mathbf{k}_1^T\left(\mathbf{v}_{k-1}+\mathbf{B}_{k-1}\mathbf{x}_{k-1}-\mathbf{f}_{k-1}\right) \\ &-2\mathbf{k}_2^T\left(\mathbf{v}_k+\mathbf{B}_k\mathbf{x}_k-\mathbf{f}_k\right) \\ &-2\mathbf{k}_3^T\left(\mathbf{x}_k-\mathbf{T}\mathbf{x}_{k-1}-\mathbf{v}_m\right)\end{aligned} \tag{118}$$

Note that there are three quadratic forms $\left(\mathbf{v}^T\mathbf{W}\mathbf{v}\right)$ to be minimized subject to three constraint equations. φ can be minimized by using Lagrange's method of undetermined multipliers $\mathbf{k}_1, \mathbf{k}_2, \mathbf{k}_3$ and setting the partial derivatives of (118) to zero.

$$\frac{\partial\varphi}{\partial\mathbf{v}_{k-1}} = 2\mathbf{v}_{k-1}^T\mathbf{W}_{k-1} - 2\mathbf{k}_1^T = \mathbf{0} \qquad \frac{\partial\varphi}{\partial\mathbf{v}_k} = 2\mathbf{v}_k^T\mathbf{W}_k - 2\mathbf{k}_2^T = \mathbf{0}$$

$$\frac{\partial\varphi}{\partial\mathbf{v}_m} = 2\mathbf{v}_m^T\mathbf{W}_m + 2\mathbf{k}_3^T = \mathbf{0} \qquad \frac{\partial\varphi}{\partial\mathbf{x}_{k-1}} = -2\mathbf{k}_1^T\mathbf{B}_{k-1} + 2\mathbf{k}_3^T\mathbf{T} = \mathbf{0}$$

$$\frac{\partial\varphi}{\partial\mathbf{x}_k} = -2\mathbf{k}_2^T\mathbf{B}_k - 2\mathbf{k}_3^T = \mathbf{0}$$

Transposing, re-arranging and dividing each equation by 2 gives

$$\begin{aligned}\mathbf{W}_{k-1}\mathbf{v}_{k-1} - \mathbf{k}_1 &= \mathbf{0} \\ \mathbf{W}_k\mathbf{v}_k - \mathbf{k}_2 &= \mathbf{0} \\ \mathbf{W}_m\mathbf{v}_m + \mathbf{k}_3 &= \mathbf{0} \\ -\mathbf{B}_{k-1}^T\mathbf{k}_1 + \mathbf{T}^T\mathbf{k}_3 &= \mathbf{0} \\ -\mathbf{B}_{k-1}^T\mathbf{k}_2 - \mathbf{k}_3 &= \mathbf{0}\end{aligned}$$

These five equations together with the primary and secondary models form a system of normal equations that can be written in the form of a hyper-matrix

$$\left[\begin{array}{ccc|ccccc} \mathbf{W}_{k-1} & \mathbf{0} & \mathbf{0} & -\mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{W}_k & \mathbf{0} & \mathbf{0} & -\mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{W}_m & \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{0} & \mathbf{0} & -\mathbf{B}_{k-1}^T & \mathbf{0} & \mathbf{T}^T & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & -\mathbf{B}_k^T & -\mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{B}_{k-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{B}_k \\ \mathbf{0} & \mathbf{0} & -\mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & -\mathbf{T} & \mathbf{I} \end{array}\right] \left[\begin{array}{c} \mathbf{v}_{k-1} \\ \mathbf{v}_k \\ \mathbf{v}_m \\ \hline \mathbf{k}_1 \\ \mathbf{k}_2 \\ \mathbf{k}_3 \\ \mathbf{x}_{k-1} \\ \mathbf{x}_k \end{array}\right] = \left[\begin{array}{c} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \hline \mathbf{0} \\ \mathbf{0} \\ \mathbf{f}_{k-1} \\ \mathbf{f}_k \\ \mathbf{0} \end{array}\right] \tag{119}$$

Partial Solution $\mathbf{x}'_{k-1}$

Using only the observations $\mathbf{l}_{k-1}$ at time t_{k-1} a partial solution for $\mathbf{x}_{k-1}$ designated $\mathbf{x}'_{k-1}$ can be obtained from (119) by deleting all matrices associated with the primary model at t_k and the secondary model. This gives

$$\begin{bmatrix} \mathbf{W}_{k-1} & -\mathbf{I} & \mathbf{0} \\ \mathbf{0} & -\mathbf{B}_{k-1}^T & \mathbf{0} \\ \mathbf{I} & \mathbf{0} & \mathbf{B}_{k-1} \end{bmatrix} \begin{bmatrix} \mathbf{v}'_{k-1} \\ \mathbf{k}'_1 \\ \mathbf{x}'_{k-1} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{f}_{k-1} \end{bmatrix}$$

Changing sign where appropriate and re-arranging gives

$$\begin{bmatrix} -\mathbf{W}_{k-1} & \mathbf{I} & \mathbf{0} \\ \mathbf{I} & \mathbf{0} & \mathbf{B}_{k-1} \\ \mathbf{0} & \mathbf{B}_{k-1}^T & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{v}'_{k-1} \\ \mathbf{k}'_1 \\ \mathbf{x}'_{k-1} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{f}_{k-1} \\ \mathbf{0} \end{bmatrix} \tag{120}$$

Equation (120) is identical in form to (13) and the partial solution is given by (36) to (38)

$$\mathbf{x}'_{k-1} = \mathbf{N}_{k-1}^{-1}\mathbf{t}_{k-1} \tag{121}$$

where

$$\mathbf{N}_{k-1} = \mathbf{B}_{k-1}^T\mathbf{W}_{k-1}\mathbf{B}_{k-1} \qquad \mathbf{t}_{k-1} = \mathbf{B}_{k-1}^T\mathbf{W}_{k-1}\mathbf{f}_{k-1} \tag{122}$$

Cofactor Matrix of Partial Solution $\mathbf{Q}_{x'_{k-1}}$

The cofactor matrix of the partial solution is given by (42) as

$$\mathbf{Q}_{x'_{k-1}} = \mathbf{N}_{k-1}^{-1} \tag{123}$$

Solution for $\hat{\mathbf{x}}_k$

Considering the partitioned form of the normal equations (119) we can obtain a solution for $\hat{\mathbf{x}}_k$ using <u>all</u> the information in the primary models at t_{k-1} and t_k and the secondary model. This is achieved by using the reduction process set out previously [see equations (13) to (19)]. First eliminate $\mathbf{v}_{k-1}, \mathbf{v}_k$ and $\mathbf{v}_m$ by applying (16) to (119). After some simplification we arrive at

$$\begin{bmatrix} -\mathbf{B}_{k-1}^T & \mathbf{0} & \mathbf{T}^T & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -\mathbf{B}_k^T & -\mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{Q}_{k-1} & \mathbf{0} & \mathbf{0} & \mathbf{B}_{k-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{Q}_k & \mathbf{0} & \mathbf{0} & \mathbf{B}_k \\ \mathbf{0} & \mathbf{0} & \mathbf{Q}_m & -\mathbf{T} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{k}_1 \\ \mathbf{k}_2 \\ \mathbf{k}_3 \\ \mathbf{x}_{k-1} \\ \mathbf{x}_k \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{f}_{k-1} \\ \mathbf{f}_k \\ \mathbf{0} \end{bmatrix} \tag{124}$$

and performing elementary row and column transformations on (124) gives a set of reduced normal equations where $\mathbf{v}_{k-1}, \mathbf{v}_k$ and $\mathbf{v}_m$ have been eliminated.

$$\left[\begin{array}{c|cccc} \mathbf{Q}_{k-1} & \mathbf{B}_{k-1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \hline -\mathbf{B}_{k-1}^T & \mathbf{0} & \mathbf{T} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -\mathbf{T} & \mathbf{Q}_m & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & -\mathbf{I} & \mathbf{0} & -\mathbf{B}_k^T \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{B}_k & \mathbf{Q}_k \end{array}\right] \left[\begin{array}{c} \mathbf{k}_1 \\ \hline \mathbf{x}_{k-1} \\ \mathbf{k}_3 \\ \delta\mathbf{x}_k \\ \mathbf{k}_2 \end{array}\right] = \left[\begin{array}{c} \mathbf{f}_{k-1} \\ \hline \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{f}_k \end{array}\right] \tag{125}$$

Now, $\mathbf{k}_1$ is eliminated from (125) by applying (16) and simplified using the auxiliaries $\mathbf{N}_{k-1}$ and $\mathbf{t}_{k-1}$ in (122) to give

$$\left[\begin{array}{c|ccc} \mathbf{N}_{k-1} & \mathbf{T}^T & \mathbf{0} & \mathbf{0} \\ \hline -\mathbf{T} & \mathbf{Q}_m & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & -\mathbf{I} & \mathbf{0} & -\mathbf{B}_k^T \\ \mathbf{0} & \mathbf{0} & \mathbf{B}_k & \mathbf{Q}_k \end{array}\right] \left[\begin{array}{c} \mathbf{x}_{k-1} \\ \hline \mathbf{k}_3 \\ \mathbf{x}_k \\ \mathbf{k}_2 \end{array}\right] = \left[\begin{array}{c} \mathbf{t}_{k-1} \\ \hline \mathbf{0} \\ \mathbf{0} \\ \mathbf{f}_k \end{array}\right] \tag{126}$$

Now $\mathbf{x}_{k-1}$ is eliminated from (126) by applying (16) and simplified to give a reduced set of normal equations containing only variates $(\mathbf{x}_k, \mathbf{k}_2, \mathbf{k}_3)$ related to the primary model at t_k and the secondary model.

$$\left[\begin{array}{c|cc} \mathbf{Q}_m + \mathbf{T}\mathbf{N}_{k-1}^{-1}\mathbf{T}^T & \mathbf{I} & \mathbf{0} \\ \hline -\mathbf{I} & \mathbf{0} & -\mathbf{B}_k^T \\ \mathbf{0} & \mathbf{B}_k & \mathbf{Q}_k \end{array}\right] \left[\begin{array}{c} \mathbf{k}_3 \\ \hline \mathbf{x}_k \\ \mathbf{k}_2 \end{array}\right] = \left[\begin{array}{c} \mathbf{T}\mathbf{N}_{k-1}^{-1}\mathbf{t}_{k-1} \\ \hline \mathbf{0} \\ \mathbf{f}_k \end{array}\right] \tag{127}$$

Predicted Value of Parameters $\mathbf{x}'_k$ (Predicted State Vector)

The normal equations (127) can be simplified by introducing the <u>predicted value of the parameters</u> at t_k, designated $\mathbf{x}'_k$ and its <u>cofactor matrix</u> $\mathbf{Q}_{x'_k}$. In Kalman Filtering these are known as the <u>predicted state vector</u> and the <u>predicted state cofactor matrix</u> respectively.

Following equations (113) we write the predicted state vector $\mathbf{x}'_k = \mathbf{T}\mathbf{x}'_{k-1}$ and substituting (121) gives

$$\mathbf{x}'_k = \mathbf{T}\mathbf{N}_{k-1}^{-1}\mathbf{t}_{k-1} \tag{128}$$

Cofactor Matrix of Predicted Parameters $\mathbf{Q}_{x'_k}$ (Predicted State Cofactor Matrix)

Applying cofactor propagation to (128) gives

$$\mathbf{Q}_{x'_k} = \left(\mathbf{T}\mathbf{N}_{k-1}^{-1}\right)\mathbf{Q}_{\mathrm{t}_{k-1}}\left(\mathbf{T}\mathbf{N}_{k-1}^{-1}\right)^T = \mathbf{T}\mathbf{N}_{k-1}^{-1}\mathbf{Q}_{\mathrm{t}_{k-1}}\mathbf{N}_{k-1}^{-1}\mathbf{T}^T \tag{129}$$

The cofactor matrix $\mathbf{Q}_{\mathrm{t}_{k-1}}$ can be obtained by applying cofactor propagation to (122) as

$$\begin{aligned} \mathbf{Q}_{\mathrm{t}_{k-1}} &= \left(\mathbf{B}_{k-1}^T\mathbf{W}_{k-1}\right)\mathbf{Q}_{\mathrm{f}_{k-1}}\left(\mathbf{B}_{k-1}^T\mathbf{W}_{k-1}\right)^T \\ &= \mathbf{B}_{k-1}^T\mathbf{W}_{k-1}\mathbf{Q}_{k-1}\mathbf{W}_{k-1}\mathbf{B}_{k-1} \\ &= \mathbf{B}_{k-1}^T\mathbf{W}_{k-1}\mathbf{B}_{k-1} \\ &= \mathbf{N}_{k-1} \end{aligned} \tag{130}$$

Substituting (130) into the right-hand side of (129) gives

$$\mathbf{Q}_{x'_k} = \mathbf{T}\mathbf{N}_{k-1}^{-1}\mathbf{N}_{k-1}\mathbf{N}_{k-1}^{-1}\mathbf{T}^T = \mathbf{T}\mathbf{N}_{k-1}^{-1}\mathbf{T}$$

By inspection of (113) we see that any cofactor propagation involving the secondary model must include the contribution of the model errors, thus the cofactor matrix for the predicted parameters is

*** $$\mathbf{Q}_{x'_k} = \mathbf{T}\mathbf{N}_{k-1}^{-1}\mathbf{T}^T + \mathbf{Q}_m = \mathbf{T}\mathbf{Q}_{x'_{k-1}}\mathbf{T} + \mathbf{Q}_m \tag{131}$$

[Note that applying cofactor propagation to the dynamic model in equations (113) gives $\mathbf{Q}_{x_k} = \mathbf{T}\mathbf{Q}_{x_{k-1}}\mathbf{T} + \mathbf{Q}_m$]

The quantities in (128) and (131) appear in (127) which allows (127) to be written as

$$\left[\begin{array}{c|cc} \mathbf{Q}_{x'_k} & \mathbf{I} & \mathbf{0} \\ \hline -\mathbf{I} & \mathbf{0} & -\mathbf{B}_k^T \\ \mathbf{0} & \mathbf{B}_k & \mathbf{Q}_k \end{array}\right] \left[\begin{array}{c} \mathbf{k}_3 \\ \hline \mathbf{x}_k \\ \mathbf{k}_2 \end{array}\right] = \left[\begin{array}{c} \mathbf{x}'_k \\ \hline \mathbf{0} \\ \mathbf{f}_k \end{array}\right] \tag{132}$$

Solution for Parameters $\hat{\mathbf{x}}_k$ (Filtered State Vector)

In Kalman Filtering $\hat{\mathbf{x}}_k$ is known as the filtered state vector. To solve for $\mathbf{x}_k$ and thus $\hat{\mathbf{x}}_k$, we may eliminate $\mathbf{k}_3$ from (132) by applying (16) giving

$$\begin{bmatrix} \mathbf{W}_{x'_k} & -\mathbf{B}_k^T \\ \mathbf{B}_k & \mathbf{Q}_k \end{bmatrix} \begin{bmatrix} \mathbf{x}_k \\ \mathbf{k}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{W}_{x'_k}\mathbf{x}'_k \\ \mathbf{f}_k \end{bmatrix} \tag{133}$$

The first equation in (133) is $\mathbf{W}_{x'_k}\mathbf{x}_k - \mathbf{B}_k^T\mathbf{k}_2 = \mathbf{W}_{x'_k}\mathbf{x}'_k$ and pre-multiplying both sides by $\mathbf{Q}_{x'_k}$ gives

$$\mathbf{x}_k = \mathbf{x}'_k + \mathbf{Q}_{x'_k}\mathbf{B}_k^T\mathbf{k}_2 \tag{134}$$

$\mathbf{k}_2$ is now obtained by applying (16) to (133) giving

$$\mathbf{k}_2 = \left(\mathbf{Q}_k + \mathbf{B}_k\mathbf{Q}_{x'_k}\mathbf{B}_k^T\right)^{-1}\left(\mathbf{f}_k - \mathbf{B}_k\mathbf{x}'_k\right) \tag{135}$$

and substituting (135) into (134) gives

$$\mathbf{x}_k = \mathbf{x}'_k + \mathbf{Q}_{x'_k}\mathbf{B}_k^T\left(\mathbf{Q}_k + \mathbf{B}_k\mathbf{Q}_{x'_k}\mathbf{B}_k^T\right)^{-1}\left(\mathbf{f}_k - \mathbf{B}_k\mathbf{x}'_k\right) \tag{136}$$

In Kalman Filtering, the expression

*** $$\mathbf{K} = \mathbf{Q}_{x'_k}\mathbf{B}_k^T\left(\mathbf{Q}_k + \mathbf{B}_k\mathbf{Q}_{x'_k}\mathbf{B}_k^T\right)^{-1} \tag{137}$$

is called the gain matrix and substituting (137) into (136) gives

*** $$\mathbf{x}_k = \mathbf{x}'_k + \mathbf{K}\left(\mathbf{f}_k - \mathbf{B}_k\mathbf{x}'_k\right) \tag{138}$$

The term in parentheses on the right-hand side of (138) is known as the vector of predicted residuals $\mathbf{v}'_k$ and

$$\mathbf{v}'_k = \mathbf{f}_k - \mathbf{B}_k\mathbf{x}'_k \tag{139}$$

We now have a solution for $\hat{\mathbf{x}}_k$

$$\hat{\mathbf{x}}_k = \mathbf{x}'_k + \mathbf{K}\mathbf{v}'_k \tag{140}$$

In Kalman Filtering $\hat{\mathbf{x}}_k$ is known as the filtered state vector.

The Cofactor Matrix $\mathbf{Q}_{\hat{x}_k}$ (Filtered State Cofactor Matrix)

In Kalman Filtering, the cofactor matrix of the parameters $\hat{\mathbf{x}}_k$ (the filtered state vector) is known as the filtered state cofactor matrix. It can be determined by using cofactor propagation as follows. Re-arrange (138) as

$$\begin{aligned}\mathbf{x}_k &= \mathbf{x}'_k + \mathbf{K}\left(\mathbf{f}_k - \mathbf{B}_k\mathbf{x}'_k\right)\\ &= \mathbf{I}\mathbf{x}'_k + \mathbf{K}\mathbf{f}_k - \mathbf{K}\mathbf{B}_k\mathbf{x}'_k\\ &= \left(\mathbf{I} - \mathbf{K}\mathbf{B}_k\right)\mathbf{x}'_k + \mathbf{K}\mathbf{f}_k\end{aligned}$$

and apply cofactor propagation to give

$$\mathbf{Q}_{\hat{x}_k} = \left(\mathbf{I} - \mathbf{K}\mathbf{B}_k\right)\mathbf{Q}_{x'_k}\left(\mathbf{I} - \mathbf{K}\mathbf{B}_k\right)^T + \mathbf{K}\mathbf{Q}_{\mathrm{f}_k}\mathbf{K}^T \tag{141}$$

Now, since the numeric terms in $\mathbf{f}_k$ will be functions of the observations $\mathbf{l}_k$ (and some constants) we may write $\mathbf{Q}_{\mathrm{f}_k} = \mathbf{Q}_k$ and substituting this result into (141) gives

$$\mathbf{Q}_{\hat{x}_k} = \left(\mathbf{I} - \mathbf{K}\mathbf{B}_k\right)\mathbf{Q}_{x'_k}\left(\mathbf{I} - \mathbf{K}\mathbf{B}_k\right)^T + \mathbf{K}\mathbf{Q}_k\mathbf{K}^T \tag{142}$$

In Kalman Filtering, the term $\left(\mathbf{I} - \mathbf{K}\mathbf{B}_k\right)$ is known as the cofactor update matrix.

Expanding and gathering terms in (142) gives

$$\mathbf{Q}_{\hat{x}_k} = \mathbf{Q}_{x'_k} - \mathbf{K}\mathbf{B}_k\mathbf{Q}_{x'_k} + \left(\mathbf{K}\mathbf{Q}_k + \mathbf{K}\mathbf{B}_k\mathbf{Q}_{x'_k}\mathbf{B}_k^T - \mathbf{Q}_{x'_k}\mathbf{B}_k^T\right)\mathbf{K}^T \tag{143}$$

The term in parenthesis on the right-hand side of (143) is equal to **0**, which can be proved by re-arranging (137). Hence, we may express the cofactor matrix of the parameters at time t_k as

***
$$\mathbf{Q}_{\hat{x}_k} = \left(\mathbf{I} - \mathbf{K}\mathbf{B}_k\right)\mathbf{Q}_{x'_k} \tag{144}$$

In practical applications of Kalman Filtering, the cofactor update matrix $\left(\mathbf{I} - \mathbf{K}\mathbf{B}_k\right)$ can sometimes become 'unstable' due to small rounding errors in the computation. To contain the instability, (142) is preferred over (144) when computing the filtered state cofactor matrix.

Summary of Kalman Filter Equations

With initial estimates of the state vector $\hat{\mathbf{x}}_{k-1}$ and the state cofactor matrix $\mathbf{Q}_{\hat{x}_{k-1}}$; and with the cofactor matrix of the dynamic model $\mathbf{Q}_m$ a Kalman Filter has the following five general steps

(1) Compute the predicted state vector at t_k

$$\mathbf{x}'_k = \mathbf{T}\hat{\mathbf{x}}_{k-1}$$

(2) Compute the predicted state cofactor matrix at t_k

$$\mathbf{Q}_{x'_k} = \mathbf{T}\mathbf{Q}_{\hat{x}_{k-1}}\mathbf{T} + \mathbf{Q}_m$$

(3) Compute the Kalman *Gain* matrix

$$\mathbf{K} = \mathbf{Q}_{x'_k}\mathbf{B}_k^T\left(\mathbf{Q}_k + \mathbf{B}_k\mathbf{Q}_{x'_k}\mathbf{B}_k^T\right)^{-1}$$

(4) Compute the filtered state vector by updating the predicted state with the measurements at t_k

$$\mathbf{x}_k = \mathbf{x}'_k + \mathbf{K}\left(\mathbf{f}_k - \mathbf{B}_k\mathbf{x}'_k\right)$$

(5) Compute the filtered state cofactor matrix

$$\mathbf{Q}_{\hat{x}_k} = \left(\mathbf{I} - \mathbf{K}\mathbf{B}_k\right)\mathbf{Q}_{x'_k}$$

Go to step (1) and repeat the process for the next measurement epoch t_{k+1}

The Kalman filter equations are relatively easy to implement on modern computers (a reason for its popularity) and the examples studied below will be supplemented by MATLAB[4] computer code in the appendix.

[4] MATLAB, a registered trademark of The MathWorks, Inc., is a high-performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation.

Example 5: Determination of a Theoretical Distance by an EDM using a Kalman Filter

The EDM component of a Total Station measures distances by phase comparison of an emitted and reflected modulated light beam. The measurement is an electro/optical process and the distance we see displayed after pressing the measure button on the Total Station is the "filtered" value of many hundreds of individual measurements, since a measurement takes only a number of milliseconds. This value is the result of a Kalman filter process.

Consider the following sequence of measurements at times $t_1, t_2, t_3 \dots$ etc ,

355.425, 355.438, 355.397, 355.429, 355.423, …

The variation in the measurements is assumed to be due to normally distributed random errors arising from the internal measurement process; often called the process noise. [The measurement sequence above, was generated by adding normally distributed random errors with mean zero and standard deviation 0.010 m to a constant value of 355.420 m.]

How will a Kalman filter produce the "filtered" value from this sequence?

First, let us assume that the measurement model is

$$\mathbf{l}_k + \mathbf{v}_k = \hat{\mathbf{l}}_k \qquad (145)$$

where $\mathbf{l}_k$ is the $n,1$ vector of measurements, $\mathbf{v}_k$ is the $n,1$ vector of residuals (small unknown corrections to the measurements) and $\hat{\mathbf{l}}_k$ are estimates of the true (but unknown) value of the measurements. n is the number of measurements, that in this case is one. The primary measurement model can be expressed in terms of the filtered state vector $\hat{\mathbf{x}}_k$ at time t_k as

$$\mathbf{v}_k + \mathbf{B}_k \hat{\mathbf{x}}_k = \mathbf{f}_k \qquad (146)$$

In this case $\hat{\mathbf{x}}_k$ contains the elements of $\hat{\mathbf{l}}_k$, both vectors containing single quantities and $\mathbf{f}_k = -\mathbf{l}_k$ also both containing single quantities (the measured distance at t_k). The matrix **B** will contain a single quantity, $\mathbf{B} = [-1]$.

Secondly, the dynamic model linking the elements of the state vector at times t_{k-1} and t_k is

$$\mathbf{x}_k = \mathbf{T}\mathbf{x}_{k-1} + \mathbf{v}_m \qquad (147)$$

The state vector contains a single element that should remain unchanged between t_{k-1} and t_k (any change is simply due to measurement errors) then the transition matrix **T** will contain a single element $\mathbf{T} = [1]$ and there are no assumed corrections to this model; hence $\mathbf{v}_m = \mathbf{0}$, $\mathbf{H} = \mathbf{0}$, $\mathbf{w} = \mathbf{0}$ and from equation (115) $\mathbf{Q}_m = \mathbf{0}$.

Lastly, an estimate of the cofactor matrix of the measurements **Q** and the filtered state cofactor matrix $\mathbf{Q}_{x_1}$ must be made. Let us assume (guess) that the measurements have a standard deviation of 10 mm (0.010 m) and hence their estimated variance is $(0.010)^2$ and $\mathbf{Q} = \left[(0.010)^2\right]$.

Since our primary measurement model has a state vector containing a single value (the measurement), then $\mathbf{Q}_x$ will only contain a single value, and we have as a starting estimate $\mathbf{Q}_{x_1} = \left[(0.010)^2\right]$, the same as $\mathbf{Q}$.

Now we can now start the Kalman filter at epoch t_2 using the values at t_1 as filtered estimates.

---

(1) Compute the predicted state vector at epoch t_2 using the measurement 355.425 at t_1 as the filtered state $\hat{\mathbf{x}}_1$

$$\begin{aligned}\mathbf{x}'_2 &= \mathbf{T}\hat{\mathbf{x}}_1\\ &= [1][355.425]\\ &= 355.425\end{aligned}$$

(2) Compute the predicted state cofactor matrix at t_2 using $\mathbf{Q}_{x_1} = \left[(0.010)^2\right]$ as the filtered estimate

$$\begin{aligned}\mathbf{Q}_{x'_2} &= \mathbf{T}\mathbf{Q}_{x_1}\mathbf{T}^T + \mathbf{Q}_m\\ &= [1]\left[(0.010)^2\right][1] + 0\\ &= (0.010)^2\end{aligned}$$

(3) Compute the Kalman *Gain matrix* noting that $\mathbf{Q} = \left[(0.010)^2\right]$

$$\begin{aligned}\mathbf{K} &= \mathbf{Q}_{x'_2}\mathbf{B}_2^T\left(\mathbf{Q} + \mathbf{B}_2\mathbf{Q}_{x'_2}\mathbf{B}_2^T\right)^{-1}\\ &= \left[(0.010)^2\right][-1]\\ &\quad \times\left(\left[(0.010)^2\right] + [-1]\left[(0.010)^2\right][-1]\right)^{-1}\\ &= \left[-(0.010)^2\right]\left[2(0.010)^2\right]^{-1}\\ &= -0.500\end{aligned}$$

(4) Compute the filtered state vector $\hat{\mathbf{x}}_2$ by updating the predicted state with the measurements at t_2

$$\begin{aligned}\hat{\mathbf{x}}_2 &= \mathbf{x}'_2 + \mathbf{K}\left(\mathbf{f}_2 - \mathbf{B}_2\mathbf{x}'_2\right)\\ &= [355.425] + \{[-0.500]\\ &\quad \times([-355.438] - [-1][355.425])\}\\ &= 355.4315\end{aligned}$$

(5) Compute the filtered state cofactor matrix at t_2

$$\begin{aligned}\mathbf{Q}_{x_2} &= \left(\mathbf{I} - \mathbf{K}_2\mathbf{B}_2\right)\mathbf{Q}_{x'_2}\\ &= \left([1] - [-0.500][-1]\right)\left[(0.010)^2\right]\\ &= 0.000050\end{aligned}$$

Go to step **(1)** and repeat the process for the next measurement epoch t_3.

The values from the Kalman filter for epochs t_3, t_4 and t_5 are

epoch t_3	epoch t_4	epoch t_5
$\mathbf{x}'_3 = 355.4315$	$\mathbf{x}'_4 = 355.4200$	$\mathbf{x}'_5 = 355.4223$
$\mathbf{Q}_{x'_3} = 0.000050$	$\mathbf{Q}_{x'_4} = 0.000033$	$\mathbf{Q}_{x'_5} = 0.000025$
$\mathbf{K}_3 = -0.333333$	$\mathbf{K}_4 = -0.250000$	$\mathbf{K}_5 = -0.200000$
$\hat{\mathbf{x}}_3 = 355.4200$	$\hat{\mathbf{x}}_4 = 355.4223$	$\hat{\mathbf{x}}_5 = 355.4224$
$\mathbf{Q}_{\hat{x}_3} = 0.000033$	$\mathbf{Q}_{\hat{x}_4} = 0.000025$	$\mathbf{Q}_{\hat{x}_5} = 0.000020$

So, the sequence of measurements is

355.425, 355.438, 355.397, 355.429, 355.423, …

and the Kalman filter estimates $\hat{\mathbf{x}}$ (the filtered values) are

355.425, 355.4315, 355.4200, 355.4223, 355.4224, …

Something that should be noted is that the filtered state cofactor matrix $\mathbf{Q}_x$ contains the estimate of the variance of the filtered value (variance is standard deviation squared). We started with an estimated value $\mathbf{Q}_{x_1} = \mathbf{Q} = (0.010)^2 = 0.000100$ and after five epochs the estimated value had reduced to $\mathbf{Q}_{x_5} = 0.000020$ equivalent to an estimated standard deviation of 0.0045 m. So the Kalman filter gives estimates with a better precision than the assumed precision of the measurement sequence; as we should expect from a least squares process. The Kalman filter for this simple case is very easily programmed and Appendix 1 contains a Kalman filter program (*edm.m*) written in the MATLAB language that processes 250 EDM measurements. These measurements are obtained by adding normally distributed random errors, with mean zero and standard deviation 0.010 m, to a constant value 355.420 m. Figure 1 below shows two plots (i) the filtered estimate of the distance (the filtered state) as a black solid line and the 250 measurements as black dots and (ii) a plot of the standard deviation of the filtered distance. After processing the 250 measurements the filtered distance was 355.4199 m with a standard deviation of 0.000632 m.

It is interesting to note that if all 250 measurements $x_1, x_2, x_3, \ldots, x_{250}$ (each with standard deviation $s_x = 0.010$ m) had been recorded and the mean $\bar{x} = \dfrac{x_1 + x_2 + \cdots + x_{250}}{250}$ computed, then propagation of variances gives the standard deviation of the mean as $s_{\bar{x}} = \dfrac{s_x}{\sqrt{250}} = 0.000632$ m which is the same as the Kalman filter result.

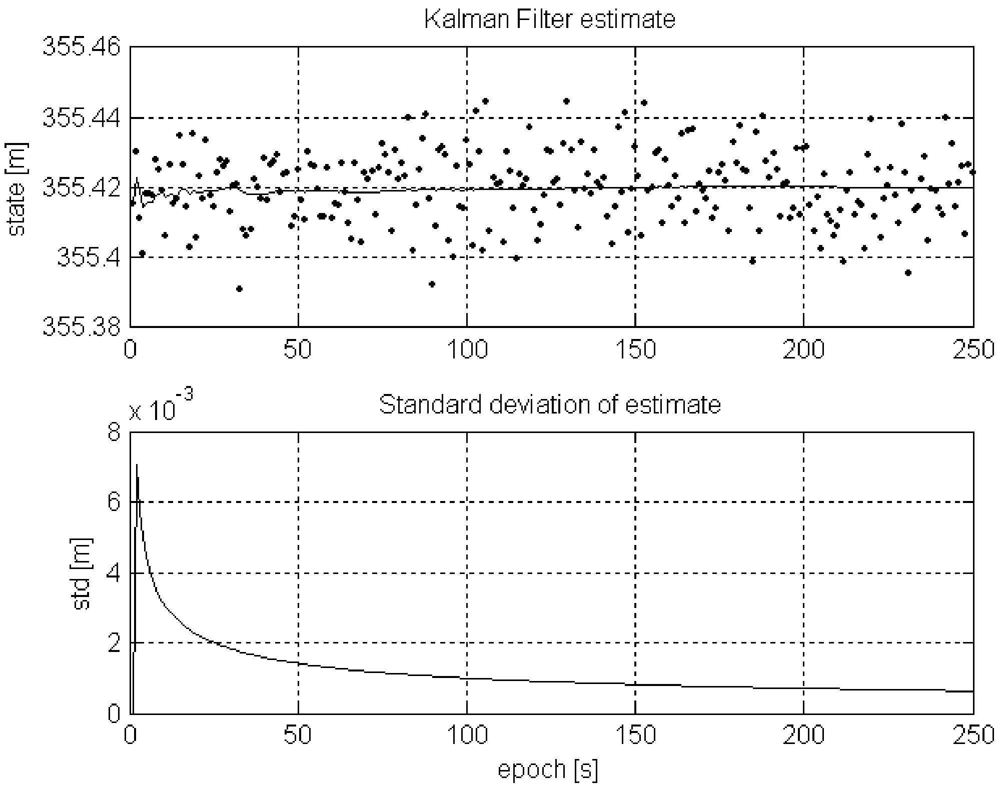
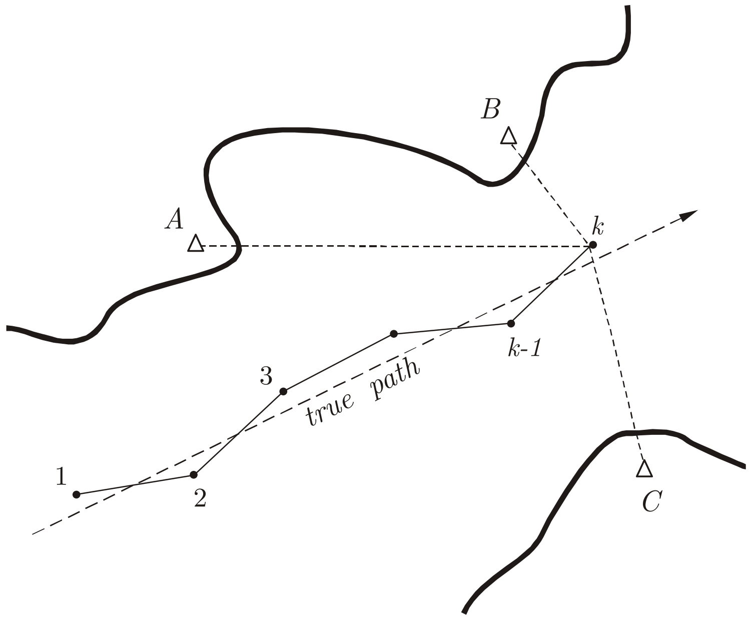


Figure 11. MATLAB plots of filtered state standard deviation of edm distance.

Example 6: Determination of Position and Velocity of a Ship in a Navigation Channel.

Figure 12 shows the path of a ship in a navigation channel as it moves down the shipping channel at a constant heading and speed. Navigation equipment on board automatically measures distances to transponders at three navigation beacons *A, B* and *C* at 60-second intervals. The measured distances are known to have a standard deviation of 1 metre and the solid line in Figure 2 represents solutions of the ship's position for each set of measurements at the 60-second time intervals. The true path of the ship is shown as the dotted line.



The coordinates of the three navigation beacons are:

A : 10000.000 E
10000.000 N

B : 13880.000 E
11250.000 N

C : 15550.000 E
7160.000 N

Figure 12. Path of a ship in a navigation channel.

The transponder measurements, from the ship to the navigation beacons *A, B* and *C* at 60-second time intervals are shown in Table 1 below. The data were generated by assuming the starting coordinates of the ship were 7875.000 m East and 6319.392 m North and the ship was travelling at 15 knots on a heading of 064° (1 knot = 1 nautical mile per hour and 1 nautical mile = 1852 metres). At 60-second intervals, the true ship position and distances to the beacons were computed. These distances were then "disturbed" by the addition of normally distributed random errors (with zero mean and standard deviation 1 metre) and then rounded to the nearest 0.1 m.

Measurement Epoch	Transponder measurements to navigation beacons		
	A	B	C
1	4249.7	7768.6	7721.1
2	3876.1	7321.4	7288.5
3	3518.4	6872.2	6857.6
4	3193.3	6426.0	6429.1
5	2903.6	5982.6	6009.7
6	2664.0	5543.2	5596.6
7	2490.9	5107.7	5191.5
8	2392.9	4678.9	4797.1
9	2383.2	4253.4	4417.8
10	2463.0	3841.7	4050.9
11	2623.2	3435.6	3709.9
12	2849.0	3054.2	3395.8
13	3126.7	2692.9	3119.4
14	3446.9	2366.6	2891.1
15	3793.4	2096.4	2724.4
16	4166.0	1900.6	2630.9
17	4552.2	1804.7	2610.2
18	4956.2	1824.8	2677.4
19	5366.4	1959.6	2819.7
20	5785.0	2182.8	3023.5

Table 1. Transponder measurements at 60-second time intervals

How will a Kalman filter produce an estimated position, speed and heading of the ship from the transponder measurements?

Note that in our Kalman filter, the state vector will be $\mathbf{x}_k = \left[E_k, N_k, \dot{E}_k, \dot{N}_k\right]^T$ containing $n = 4$ elements (or parameters) where $\left(E_k, N_k\right)$ are the ship's position and $\left(\dot{E}_k, \dot{N}_k\right)$ the ship's velocity components. The speed and the heading of the ship (bearing from North) at time t_k is

$$speed = \sqrt{\left(\dot{E}_k\right)^2 + \left(\dot{N}_k\right)^2} \qquad \tan\left(heading\right) = \frac{\dot{E}_k}{\dot{N}_k} \tag{148}$$

The measurement model (primary model)

Let us assume that the primary or measurement model is

$$\mathbf{l}_k + \mathbf{v}_k = \hat{\mathbf{l}}_k \tag{149}$$

where $\mathbf{l}_k$ is the $m,1$ vector of measurements (the transponder distances), $\mathbf{v}_k$ is the $m,1$ vector of residuals (small unknown corrections to the measurements) and $\hat{\mathbf{l}}_k$ are estimates of the true (but unknown) value of the measurements. m is the number of measurements, that in this case is three at each measurement epoch. The estimates $\hat{\mathbf{l}}_k$ are non-linear functions of the coordinates *E,N* of the beacons *A, B* and *C* and the filtered state coordinates $\hat{E}_k, \hat{N}_k$ of the ship at time t_k

$$\hat{l}_j = \hat{l}\left(\hat{E}_k, \hat{N}_k, E_j, N_j\right) = \sqrt{\left(\hat{E}_k - E_j\right)^2 + \left(\hat{N}_k - N_j\right)^2} \qquad \text{for } j = A, B, C \tag{150}$$

Expanding equation (150) into a series using Taylor's theorem gives

$$\hat{l} = l' + \frac{\partial \hat{l}}{\partial \hat{E}_k}\left(\hat{E}_k - E'_k\right) + \frac{\partial \hat{l}}{\partial \hat{N}_k}\left(\hat{N}_k - N'_k\right) + \text{higher order terms}$$

where E'_k, N'_k are approximate coordinates of the ship at t_k, l' is an approximate distance computed using E'_k, N'_k and the coordinates of the beacon, and the partial derivatives are

$$\begin{aligned} \frac{\partial \hat{l}}{\partial \hat{E}_k} &= \frac{E'_k - E_j}{l'_j} = d_j \\ \frac{\partial \hat{l}}{\partial \hat{N}_k} &= \frac{N'_k - N_j}{l'_j} = c_j \qquad \text{for } j = A, B, C \end{aligned} \tag{151}$$

Re-arranging equation (149) for a single distance gives

$$v - \hat{l} = -l$$

and substituting the Taylor series approximation for $\hat{l}$ (ignoring higher-order terms) and re-arranging gives the linearized form of the primary measurement model as

$$v_j - d_j\hat{E}_k - c_j\hat{N}_k = l'_j - l_j + \left(-d_jE'_k - c_jN'_k\right) \qquad \text{for } j = A, B, C \tag{152}$$

This primary measurement model can be expressed in terms of the filtered state vector $\hat{\mathbf{x}}_k$ at time t_k in the matrix form as

$$\begin{bmatrix} v_A \\ v_B \\ v_C \end{bmatrix}_k + \begin{bmatrix} -d_A & -c_A & 0 & 0 \\ -d_B & -c_B & 0 & 0 \\ -d_C & -c_C & 0 & 0 \end{bmatrix}_k \begin{bmatrix} \hat{E} \\ \hat{N} \\ \dot{E} \\ \dot{N} \end{bmatrix}_k = \begin{bmatrix} l'_A - l_A \\ l'_B - l_B \\ l'_C - l_C \end{bmatrix}_k + \begin{bmatrix} -d_A & -c_A & 0 & 0 \\ -d_B & -c_B & 0 & 0 \\ -d_C & -c_C & 0 & 0 \end{bmatrix}_k \begin{bmatrix} E' \\ N' \\ \dot{E}' \\ \dot{N}' \end{bmatrix}_k$$

or

$$\mathbf{v}_k + \mathbf{B}_k\hat{\mathbf{x}}_k = \mathbf{l}'_k - \mathbf{l}_k + \mathbf{B}_k\mathbf{x}'_k = \mathbf{f}_k \tag{153}$$

Now in step (4) of the Kalman filter algorithm the filtered state vector $\hat{\mathbf{x}}_k$ is obtained from

$$\hat{\mathbf{x}}_k = \mathbf{x}'_k + \mathbf{K}\left(\mathbf{f}_k - \mathbf{B}_k\mathbf{x}'_k\right) \tag{154}$$

and substituting for $\mathbf{f}_k$ from equation (153) gives

$$\begin{aligned}\hat{\mathbf{x}}_k &= \mathbf{x}'_k + \mathbf{K}\left(\mathbf{l}'_k - \mathbf{l}_k + \mathbf{B}_k\mathbf{x}'_k - \mathbf{B}_k\mathbf{x}'_k\right)\\ &= \mathbf{x}'_k + \mathbf{K}\left(\mathbf{l}'_k - \mathbf{l}_k\right)\end{aligned} \tag{155}$$

Note: (i) the term $\left(\mathbf{f}_k - \mathbf{B}_k\mathbf{x}'_k\right)$ in equation (154) is often called the *predicted residuals* $\mathbf{v}'_k$ where, in our case

$$\mathbf{v}'_k = \mathbf{f}_k - \mathbf{B}_k\mathbf{x}'_k = \mathbf{l}'_k - \mathbf{l}_k \tag{156}$$

(ii) The term $\mathbf{K}\left(\mathbf{l}'_k - \mathbf{l}_k\right)$ in equation (155) is often called the *corrections to the predicted state* $\Delta\mathbf{x}$ where, in our case

$$\Delta\mathbf{x}_k = \mathbf{K}_k\left(\mathbf{l}'_k - \mathbf{l}_k\right) \tag{157}$$

The dynamic model (secondary model)

A dynamic model that is extremely simple and often used in navigation problems can be developed by considering a continuous function of time, say $y = y(t)$. Following the development by Cross (1987), we can use Taylor's theorem to expand the function $y(t)$ about the point $t = t_k$ into the series

$$y(t) = y(t_k) + (t - t_k)\,\dot{y}(t_k) + \frac{(t-t_k)^2}{2!}\ddot{y}(t_k) + \frac{(t-t_k)^3}{3!}\dddot{y}(t_k) + \cdots$$

where $\dot{y}(t_k), \ddot{y}(t_k), \dddot{y}(t_k)$, etc are derivatives of *y* with respect to *t* evaluated at $t = t_k$. Letting $t = t_k + \delta t$ and then $\delta t = t - t_k$ we may write

$$y(t_k + \delta t) = y(t_k) + \dot{y}(t_k)\,\delta t + \frac{\ddot{y}(t_k)}{2!}(\delta t)^2 + \frac{\dddot{y}(t_k)}{3!}(\delta t)^3 + \frac{\ddddot{y}(t_k)}{4!}(\delta t)^4 + \cdots \tag{158}$$

We now have power series expression for the continuous function $y(t)$ at the point $t = t_k + \delta t$ involving the function *y* and its derivatives $\dot{y},\ \ddot{y},$ etc, (all evaluated at t_k) and the time difference $\delta t = t - t_k$.

In a similar manner, if we assume $\dot{y}(t),\ \ddot{y}(t),$ etc to be continuous functions of *t* then

$$\begin{aligned}&\dot{y}(t_k + \delta t) = \dot{y}(t_k) + \ddot{y}(t_k)\,\delta t + \frac{\dddot{y}(t_k)}{2!}(\delta t)^2 + \frac{\ddddot{y}(t_k)}{3!}(\delta t)^3 + \cdots\\ &\ddot{y}(t_k + \delta t) = \ddot{y}(t_k) + \dddot{y}(t_k)\,\delta t + \frac{\ddddot{y}(t_k)}{2!}(\delta t)^2 + \cdots\\ &\dddot{y}(t_k + \delta t) = \dddot{y}(t_k) + \ddddot{y}(t_k)\,\delta t + \cdots\\ &\text{etc}\end{aligned} \tag{159}$$

Now consider two time epochs t_k and t_{k-1} separated by a time interval δt, we can combine equations (158) and (159) (with a change of subscripts for *t*) into the general matrix forms:

(i) involving terms up to $\ddot{y}$

$$\begin{bmatrix} y \\ \dot{y} \end{bmatrix}_k = \begin{bmatrix} 1 & \delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} y \\ \dot{y} \end{bmatrix}_{k-1} + \begin{bmatrix} \frac{1}{2}(\delta t)^2 \\ \delta t \end{bmatrix} [\ddot{y}]_{k-1} \tag{160}$$

(ii) involving terms up to $\dddot{y}$

$$\begin{bmatrix} y \\ \dot{y} \\ \ddot{y} \end{bmatrix}_k = \begin{bmatrix} 1 & \delta t & \frac{1}{2}(\delta t)^2 \\ 0 & 1 & \delta t \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} y \\ \dot{y} \\ \ddot{y} \end{bmatrix}_{k-1} + \begin{bmatrix} \frac{1}{6}(\delta t)^3 \\ \frac{1}{2}(\delta t)^2 \\ \delta t \end{bmatrix} [\dddot{y}]_{k-1} \tag{161}$$

(iii) involving terms up to $\ddddot{y}$

$$\begin{bmatrix} y \\ \dot{y} \\ \ddot{y} \\ \dddot{y} \end{bmatrix}_k = \begin{bmatrix} 1 & \delta t & \frac{1}{2}(\delta t)^2 & \frac{1}{6}(\delta t)^3 \\ 0 & 1 & \delta t & \frac{1}{2}(\delta t)^2 \\ 0 & 0 & 1 & \delta t \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} y \\ \dot{y} \\ \ddot{y} \\ \dddot{y} \end{bmatrix}_{k-1} + \begin{bmatrix} \frac{1}{24}(\delta t)^4 \\ \frac{1}{6}(\delta t)^3 \\ \frac{1}{2}(\delta t)^2 \\ \delta t \end{bmatrix} [\ddddot{y}]_{k-1} \tag{162}$$

In Kalman filtering and navigation problems the continuous function of time $y = y(t)$ is simply <u>position</u> so that $y(t) = \{E, N\}(t)$ where *E,N* are east and north coordinates. The derivatives are <u>velocity</u>, $\dot{y}(t) = \{\dot{E}, \dot{N}\}(t)$ (rate of change of distance), <u>acceleration</u>, $\ddot{y}(t) = \{\ddot{E}, \ddot{N}\}(t)$ (rate of change of velocity), *jerk,* $\dddot{y}(t) = \{\dddot{E}, \dddot{N}\}(t)$ (rate of change of acceleration) and rate of change of jerk, $\ddddot{y}(t) = \{\ddddot{E}, \ddddot{N}\}(t)$. In each of the cases above [equations (160), (161) and (162)], we can consider the vector on the left-hand-side of the equals sign to be the vector $\mathbf{x}_k$, the state vector, or the state the system at time t_k. The matrix on the right-hand-side is the <u>transition matrix</u> **T** and the elements of this matrix contain the links between the state vector at times t_k and t_{k-1}, i.e., $\mathbf{x}_k = \mathbf{T}\mathbf{x}_{k-1}$. The second term in the equations above is the product of two matrices (in these cases two vectors) and the result will be the vector of model residuals $\mathbf{v}_m$ (containing the same number of elements as the state vector). $\mathbf{v}_m$ is a reflection of the fact that the transition matrix does not fully describe the exact physical links between the states at times t_k and t_{k-1} and $\mathbf{v}_m = \mathbf{H}\mathbf{w}$ where **H** is a coefficient matrix and **w** is the <u>system driving noise</u>. In the equations above the system driving noise is acceleration, jerk and rate of change of jerk respectively.

We can now use these general forms to define a suitable dynamic model.

In our simple case (the ship in the channel) the state vector **x** contains four elements $\mathbf{x}_k = \left[E_k, N_k, \dot{E}_k, \dot{N}_k\right]^T$ and the appropriate dynamic model in the form of equation (160) is

$$\begin{bmatrix} E \\ N \\ \dot{E} \\ \dot{N} \end{bmatrix}_k = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} E \\ N \\ \dot{E} \\ \dot{N} \end{bmatrix}_{k-1} + \begin{bmatrix} \frac{1}{2}(\Delta t)^2 & 0 \\ 0 & \frac{1}{2}(\Delta t)^2 \\ \Delta t & 0 \\ 0 & \Delta t \end{bmatrix} \begin{bmatrix} \ddot{E} \\ \ddot{N} \end{bmatrix}_{k-1} \tag{163}$$

or

$$\mathbf{x}_k = \mathbf{T}\mathbf{x}_{k-1} + \mathbf{v}_m \tag{164}$$

where **T** is the *n,n* transition matrix and $\mathbf{v}_m$ is the $n,1$ vector of model residuals.

If we expand equation (163) we see that it is really just the matrix form of the two equations of rectilinear motion; (i) $v = u + at$ and (ii) $s = ut + \frac{1}{2}at^2$ where *s* is distance, *u* is initial velocity, *v* is final velocity, *a* is acceleration and *t* is time. In our notation they are:

(i)

$$\dot{E}_k = \dot{E}_{k-1} + \ddot{E}\,\Delta t$$
$$\dot{N}_k = \dot{N}_{k-1} + \ddot{N}\,\Delta t$$

and (ii)

$$E_k = E_{k-1} + \dot{E}_{k-1}\,\Delta t + \tfrac{1}{2}\ddot{E}\left(\Delta t\right)^2$$
$$N_k = N_{k-1} + \dot{N}_{k-1}\,\Delta t + \tfrac{1}{2}\ddot{N}\left(\Delta t\right)^2$$

The model residuals $\mathbf{v}_m = \mathbf{H}\mathbf{w}$ are

$$\begin{bmatrix} v_E \\ v_N \\ v_{\dot{E}} \\ v_{\dot{N}} \end{bmatrix} = \begin{bmatrix} \frac{1}{2}(\Delta t)^2 & 0 \\ 0 & \frac{1}{2}(\Delta t)^2 \\ \Delta t & 0 \\ 0 & \Delta t \end{bmatrix} \begin{bmatrix} \ddot{E} \\ \ddot{N} \end{bmatrix} \tag{165}$$

where the coefficient matrix **H** and the system driving noise **w** are

$$\mathbf{H} = \begin{bmatrix} \frac{1}{2}(\Delta t)^2 & 0 \\ 0 & \frac{1}{2}(\Delta t)^2 \\ \Delta t & 0 \\ 0 & \Delta t \end{bmatrix} \text{ and } \mathbf{w} = \begin{bmatrix} \ddot{E} \\ \ddot{N} \end{bmatrix} \tag{166}$$

In this simple navigation problem it is assumed that the system driving noise **w** contains small random accelerations caused by the sea and wind conditions, the steering of the ship, the engine speed variation, etc.

The cofactor matrix of the dynamic model $\mathbf{Q}_m$ is given by

$$\mathbf{Q}_m = \mathbf{H}\mathbf{Q}_w\mathbf{H}^T \tag{167}$$

where $\mathbf{Q}_w$, the cofactor matrix of the system driving noise, is

$$\mathbf{Q}_w = \begin{bmatrix} s_{\ddot{E}}^2 & 0 \\ 0 & s_{\ddot{N}}^2 \end{bmatrix}$$

and $s_{\ddot{E}}^2$, $s_{\ddot{N}}^2$ are the estimates of the variances of the accelerations in the east and north directions and the covariance is assumed to be zero. Using the coefficient matrix $\mathbf{H}$ in equation (166) we have

$$\mathbf{Q}_m = \begin{bmatrix} \frac{1}{2}(\Delta t)^2 & 0 \\ 0 & \frac{1}{2}(\Delta t)^2 \\ \Delta t & 0 \\ 0 & \Delta t \end{bmatrix} \begin{bmatrix} s_{\ddot{E}}^2 & 0 \\ 0 & s_{\ddot{N}}^2 \end{bmatrix} \begin{bmatrix} \frac{1}{2}(\Delta t)^2 & 0 & \Delta t & 0 \\ 0 & \frac{1}{2}(\Delta t)^2 & 0 & \Delta t \end{bmatrix} \tag{168}$$

Now we can now start the Kalman filter, but some initial values must be set beforehand. These will be designated (a), (b), (c) etc. and then the Kalman filter steps (1), (2), (3) etc.

(a) Set the elements of the transition matrix

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

In this exercise $\Delta t = 60\,\text{sec}$

(b) Set the cofactor matrix of the measurements

$$\mathbf{Q} = \begin{bmatrix} s_{l_A}^2 & 0 & 0 \\ 0 & s_{l_B}^2 & 0 \\ 0 & 0 & s_{l_C}^2 \end{bmatrix}$$

In this exercise $s_{l_A}^2 = s_{l_B}^2 = s_{l_C}^2 = 1.0\ \text{m}^2$

(c) Set the cofactor matrix of the system driving noise

$$\mathbf{Q}_w = \begin{bmatrix} s_{\ddot{E}}^2 & 0 \\ 0 & s_{\ddot{N}}^2 \end{bmatrix}$$

In this exercise
$s_{\ddot{E}}^2 = s_{\ddot{N}}^2 = 0.017\ \text{m}^2/\text{s}^4$

(d) Set the coefficient matrix of the system driving noise

$$\mathbf{H} = \begin{bmatrix} \frac{1}{2}(\Delta t)^2 & 0 \\ 0 & \frac{1}{2}(\Delta t)^2 \\ \Delta t & 0 \\ 0 & \Delta t \end{bmatrix}$$

(e) Compute the cofactor matrix of the dynamic model

$$\mathbf{Q}_m = \mathbf{H}\mathbf{Q}_w\mathbf{H}^T$$

(f) Set the starting estimates of the state vector. This will be the filtered state vector for epoch t_2

$$\mathbf{x}_1 = \begin{bmatrix} E \\ N \\ \dot{E} \\ \dot{N} \end{bmatrix}_1 = \begin{bmatrix} 7875.000 \text{ m} \\ 6319.392 \text{ m} \\ 7 \text{ m}/s \\ 3 \text{ m/s} \end{bmatrix}$$

(g) Set the starting estimates of the state cofactor matrix. This will be the filtered state cofactor matrix for epoch t_2

$$\mathbf{Q}_{x_k} = \begin{bmatrix} s_E^2 & 0 & 0 & 0 \\ 0 & s_N^2 & 0 & 0 \\ 0 & 0 & s_{\dot{E}}^2 & 0 \\ 0 & 0 & 0 & s_{\dot{N}}^2 \end{bmatrix}$$

In this exercise $s_E^2 = s_N^2 = 20\,\text{m}^2$ and $s_{\dot{E}}^2 = s_{\dot{N}}^2 = 0.5 \text{ m}^2/\text{s}^2$

Now start the Kalman filter at epoch t_2

(1) Compute the predicted state vector at epoch t_2 using the filtered estimate $\hat{\mathbf{x}}_1$

$$\mathbf{x}_2' = \mathbf{T}\hat{\mathbf{x}}_1$$

(2) Compute the predicted state cofactor matrix at t_2 using $\mathbf{Q}_m$ from step (e)

$$\mathbf{Q}_{x_2'} = \mathbf{T}\mathbf{Q}_{x_1}\mathbf{T}^T + \mathbf{Q}_m$$

(3) Compute the Kalman *Gain matrix*

$$\mathbf{K} = \mathbf{Q}_{x_2'}\mathbf{B}_2^T\left(\mathbf{Q} + \mathbf{B}_2\mathbf{Q}_{x_2'}\mathbf{B}_2^T\right)^{-1}$$

Using **Q** from step (b) and **B** whose elements have been determined using equations (151). The form of **B** is given in equation (153).

(4.1) Compute the numeric terms "computed – observed" distances [see equation (155)]

(4.2) Compute the filtered state vector $\hat{\mathbf{x}}_2$ by updating the predicted state [see equation (155)]

$$\hat{\mathbf{x}}_k = \mathbf{x}_k' + \mathbf{K}\left(\mathbf{l}_k' - \mathbf{l}_k\right)$$

(5) Compute the filtered state cofactor matrix at t_2

$$\mathbf{Q}_{x_2} = \left(\mathbf{I} - \mathbf{K}_2\mathbf{B}_2\right)\mathbf{Q}_{x_2'}$$

Go to step **(1)** and repeat the process for the next measurement epoch t_3.

The following output from a MATLAB program *kalship3.m* (see Appendix) processes the data in Table 1 in a Kalman Filter beginning at epoch 2.

```
>> kalship3('d:\projects\kalman\exercise\kalshipdata3.txt');


epoch =   2
Filtered State  Corrns        Filtered State cofactor matrix Qxx
E  8289.594     -5.406        1.009225 -0.797965  0.033097 -0.026169
N  6521.882     22.490       -0.797965  1.439797 -0.026169  0.047217
vE    6.823     -0.177        0.033097 -0.026169  0.506780 -0.000858
vN    3.738      0.738       -0.026169  0.047217 -0.000858  0.507243

epoch =   3
Filtered State  Corrns        Filtered State cofactor matrix Qxx
   8705.780      6.823        0.926924 -0.643621  0.030398 -0.021105
   6727.944    -18.189       -0.643621  1.218371 -0.021105  0.039955
      7.046      0.224        0.030398 -0.021105  0.494715 -0.004015
      3.141     -0.596       -0.021105  0.039955 -0.004015  0.496822

epoch =   4
Filtered State  Corrns        Filtered State cofactor matrix Qxx
   9124.759     -3.808        0.863463 -0.498398  0.028327 -0.016346
   6928.604     12.198       -0.498398  1.011477 -0.016346  0.033180
      6.922     -0.125        0.028327 -0.016346  0.483077 -0.006336
      3.541      0.400       -0.016346  0.033180 -0.006336  0.486133
:
:
:
epoch =  18
Filtered State  Corrns        Filtered State cofactor matrix Qxx
  14950.377      9.319        0.955699  0.439824  0.031474  0.014481
   9770.483      4.566        0.439824  0.822675  0.014483  0.027090
      7.134      0.307        0.031474  0.014483  0.368845  0.003650
      3.497      0.150        0.014481  0.027090  0.003650  0.371765

epoch =  19
Filtered State  Corrns        Filtered State cofactor matrix Qxx
E 15366.544    -11.869        0.732074  0.288579  0.024112  0.009501
N  9973.570     -6.708        0.288579  0.820826  0.009502  0.027033
vE    6.743     -0.391        0.024112  0.009502  0.364017  0.005200
vN    3.276     -0.221        0.009501  0.027033  0.005200  0.366470

epoch =  20
Filtered State  Corrns        Filtered State cofactor matrix Qxx
  15781.273     10.148        0.598119  0.158080  0.019704  0.005204
  10175.278      5.167        0.158080  0.847313  0.005203  0.027911
      7.077      0.334        0.019704  0.005203  0.358551  0.006055
      3.446      0.170        0.005204  0.027911  0.006055  0.361445

Filtered Values
Epoch  Distance  Velocity  Heading
  1       0.000    7.616   66.801
  2     461.400    7.779   61.286
  3     925.806    7.715   65.975
  4    1390.357    7.775   62.905
  :
  :
 18    7872.215    7.945   63.889
 19    8335.291    7.497   64.090
 20    8796.471    7.872   64.039

>>
```

Example 7: Global Warming – Estimating Trends in Temperature Anomalies.

The inspiration for this example of Kalman Filtering came from the blog of Andreas Hetland[5] (2012), An Example of Kalman Filtering: The Surface Air Temperature Anomaly. It is a case of using a Kalman Filter to determine a trend in what appears to be random data.

For this exercise we consider annual measurements of the so-called "surface air temperature anomaly" in the United States from 1880 to 2014. The US surface air temperature anomaly is the difference between the annual air temperature value and the 1951-1980 mean temperature. The data is part of the GISTEMP[6] study and is available at http://data.giss.nasa.gov/gistemp/

A plot of the data is shown in Figure 13.

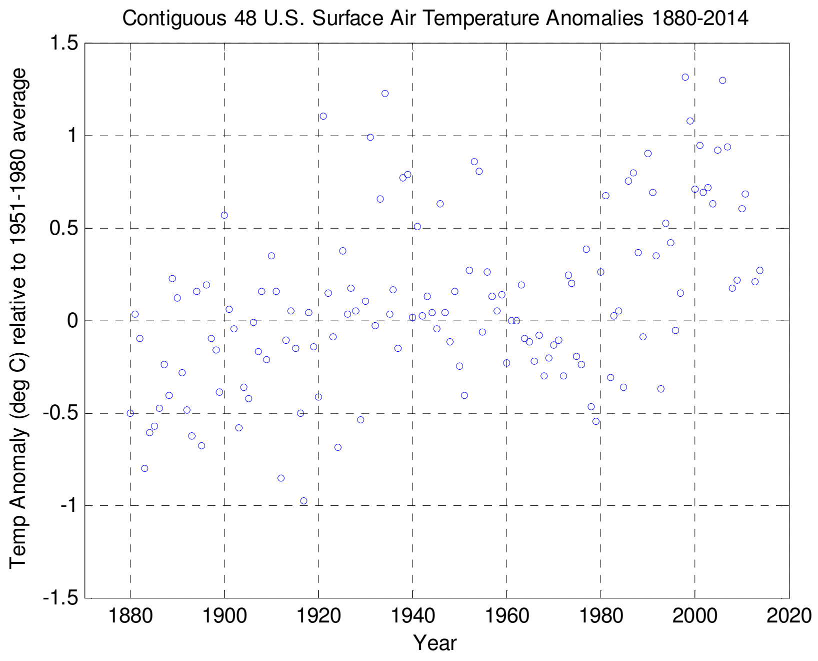


Figure 13. Temperature anomalies from 1880-2014

There does not appear to be any discernible trend in the data and the authors of the study chose to use a Centred 5-Year Moving Average as a means of uncovering a trend or signal. For a data series $x_1, x_2, \ldots, x_n$ the Centred Moving Averages y_t for the moving average period $p = 2k+1$ are

$$y_t = \frac{1}{2k+1} \sum_{j=-k}^{k} x_{t+j} \quad \text{for } t = k+1, k+2, \ldots, n-k \tag{169}$$

The moving average period p should be an odd number (for a 5-year moving average period $p = 5$ and $k = 2$).

[5] From 2011-14 Andreas Hetland was completing his PhD in Financial Econometrics at the University of Copenhagen. His thesis was titled: "On Particle Filter-Based Estimation and Inference for Dynamic Models with Unobserved Variables". He is currently (April 2015) Assistant Vice President Barclays Investment Bank, London. Unfortunately, as at September 2015, his blog seems to have disappeared.

[6] Goddard Institute for Space Studies Surface Temperature Analysis (GISTEMP). The Goddard Institute is a division of the National Aeronautics and Space Administration (NASA).

Figure 14 shows the data with the 5-Year Moving Average values as a trend line. The temperature data begin in 1880 and end in 2014 and the period for the averages is 1882-2012.

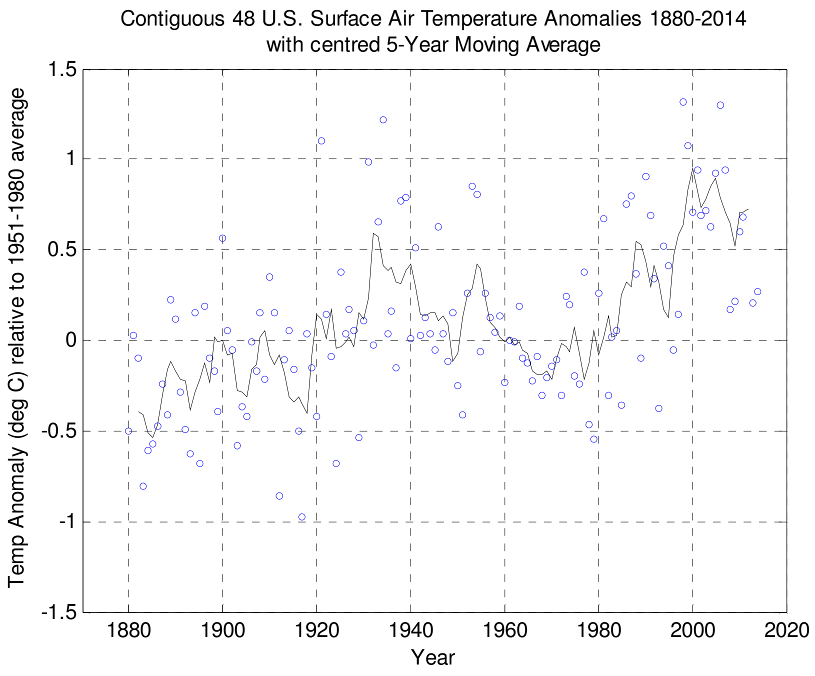


Figure 14. Temperature anomalies with 5-Year moving Average trend line

A drawback of the Centred Moving Averages is that there are no values for the last *k* years in the sequence (2013, 2014) and the trend line is not very smooth. A Kalman Filter, described below, may offer a better alternative.

The sequence of measurements (the anomalies in ^{O}C) at times $t_1, t_2, t_3, \ldots$ etc. are

$$-0.4656,\ 0.0693,\ -0.0067,\ -0.8181, \ldots$$

where $t_1 = 1880$, $t_{END} = 2014$ and there are $2014 - 1880 + 1 = 135$ measurement epochs. The measurements (affected by random errors), and residuals can be expressed as

$$\mathbf{l}_k + \mathbf{v}_k = \hat{\mathbf{l}}_k \tag{170}$$

where $\mathbf{l}_k$ is the $n,1$ vector of measurements, $\mathbf{v}_k$ is the $n,1$ vector of residuals (small unknown corrections to the measurements) and $\hat{\mathbf{l}}_k$ are estimates of the true (but unknown) value of the measurements. n is the number of measurements at each epoch, that in this case is one. The primary measurement model can be expressed in terms of the filtered state vector $\hat{\mathbf{x}}_k$ at time t_k as

$$\mathbf{v}_k + \mathbf{B}_k \hat{\mathbf{x}}_k = \mathbf{f}_k \tag{171}$$

In this case $\hat{\mathbf{x}}_k$ contains the elements of $\hat{\mathbf{l}}_k$, both vectors containing only single quantities. And $\mathbf{f}_k = -\mathbf{l}_k$ also both containing single quantities (the anomalies at t_k). The matrix **B** will contain a single quantity, $\mathbf{B} = [-1]$.

The dynamic model linking the elements of the state vector at times t_{k-1} and t_k is

$$\mathbf{x}_k = \mathbf{T}\mathbf{x}_{k-1} + \mathbf{v}_m \tag{172}$$

The state vector contains a single element that may change between t_{k-1} and t_k but the dynamics are unknown and changes will be due to measurement error and model error. The transition matrix **T** will contain a single element $\mathbf{T} = [1]$ and the model correction $\mathbf{v}_m = \mathbf{Hw}$ is a single element vector containing a random quantity. By setting $\mathbf{H} = [1]$ the model correction is equal to the system driving noise, i.e., $\mathbf{v}_m = \mathbf{w}$. We can "assign" an appropriate estimate of the standard deviation s_w and the cofactor matrix $\mathbf{Q}_w = \left[s_w^2\right]$ is now defined, and by cofactor propagation [see (115)], the cofactor matrix of the dynamic model $\mathbf{Q}_m = \mathbf{Q}_w$. We choose $s_w = 0.1$ which gives $\mathbf{Q}_w = [0.01]$

Lastly, an estimate of the cofactor matrix of the measurements **Q** and the filtered state cofactor matrix $\mathbf{Q}_{x_k}$ must be made. Let us assume (guess) that the measurements have a standard deviation of $s_l = \sqrt{0.5}$ and $\mathbf{Q} = \left[s_l^2\right] = [0.5]$. Also, since our primary measurement model has a state vector containing a single value (the measurement), then $\mathbf{Q}_x$ will only contain a single value, and we have as a starting estimate $\mathbf{Q}_{x_1} = [0.5]$, the same as **Q**.

Now we can now start the Kalman filter at epoch t_2 using the values at t_1 as filtered estimates.

---

(1) Compute the predicted state vector at epoch t_2 using the measurement -0.4656 at t_1 as the filtered state $\hat{\mathbf{x}}_1$

$$\begin{aligned}\mathbf{x}'_2 &= \mathbf{T}\hat{\mathbf{x}}_1\\ &= [1][-0.4656]\\ &= -0.4656\end{aligned}$$

(2) Compute the predicted state cofactor matrix at t_2 using $\mathbf{Q}_{x_1} = [0.5]$ as the filtered estimate

$$\begin{aligned}\mathbf{Q}_{x'_2} &= \mathbf{T}\mathbf{Q}_{x_1}\mathbf{T}^T + \mathbf{Q}_m\\ &= [1][0.5][1] + [0.01]\\ &= [0.51]\end{aligned}$$

(3) Compute the Kalman *Gain matrix* noting that $\mathbf{Q} = [0.5]$

$$\begin{aligned}\mathbf{K} &= \mathbf{Q}_{x'_2}\mathbf{B}_2^T\left(\mathbf{Q} + \mathbf{B}_2\mathbf{Q}_{x'_2}\mathbf{B}_2^T\right)^{-1}\\ &= [0.51][-1]\\ &\quad \times([0.5] + [-1][0.51][-1])^{-1}\\ &= [-0.51][1.01]^{-1}\\ &= -0.504950\end{aligned}$$

(4) Compute the filtered state vector $\hat{\mathbf{x}}_2$ by updating the predicted state with the measurements at t_2

$$\begin{aligned}\hat{\mathbf{x}}_2 &= \mathbf{x}'_2 + \mathbf{K}\left(\mathbf{f}_2 - \mathbf{B}_2\mathbf{x}'_2\right)\\ &= [-0.4656] + \{[-0.504950]\\ &\quad \times([-0.0693] - [-1][-0.4656])\}\\ &= -0.1955\end{aligned}$$

(5) Compute the filtered state cofactor matrix at t_2

$$\begin{aligned}\mathbf{Q}_{x_2} &= \left(\mathbf{I}-\mathbf{K}_2\mathbf{B}_2\right)\mathbf{Q}_{x_2'} \\ &= \left([1]-[-0.504950][-1]\right)[0.51] \\ &= 0.252476\end{aligned}$$

↰ Go to step **(1)** and repeat the process for the next measurement epoch t_3.

The values from the Kalman filter for epochs t_3, t_4 and t_5 are

epoch t_3	epoch t_4	epoch t_5
$\mathbf{x}_3' = -0.1955$	$\mathbf{x}_4' = -0.1305$	$\mathbf{x}_5' = -0.3141$
$\mathbf{Q}_{x_3'} = 0.262475$	$\mathbf{Q}_{x_4'} = 0.182121$	$\mathbf{Q}_{x_5'} = 0.143496$
$\mathbf{K}_3 = -0.344241$	$\mathbf{K}_4 = -0.266991$	$\mathbf{K}_5 = -0.222994$
$\hat{\mathbf{x}}_3 = -0.1305$	$\hat{\mathbf{x}}_4 = -0.3141$	$\hat{\mathbf{x}}_5 = -0.3788$
$\mathbf{Q}_{\hat{x}_3} = 0.172121$	$\mathbf{Q}_{\hat{x}_4} = 0.133496$	$\mathbf{Q}_{\hat{x}_5} = 0.111497$

A MATLAB function *global_warming_filter.m* (see Appendix) with a data file containing the surface air temperature anomaly for the 48 contiguous United States for 1880-2014 (derived from GISTEMP study data at: http://data.giss.nasa.gov/gistemp/graphs_v3/Fig.D.txt) gives the following output

```
>> global_warming_filter

Epoch =   2, Year = 1881, measurement =   0.0693
Filtered State  Corrn         Filtered State cofactor matrix Qxx
    -0.1955     0.2701       0.252475248

Epoch =   3, Year = 1882, measurement =  -0.0067
Filtered State  Corrn         Filtered State cofactor matrix Qxx
    -0.1305     0.0650       0.172120504

Epoch =   4, Year = 1883, measurement =  -0.8181
Filtered State  Corrn         Filtered State cofactor matrix Qxx
    -0.3141    -0.1836       0.133495843

Epoch =   5, Year = 1884, measurement =  -0.6041
Filtered State  Corrn         Filtered State cofactor matrix Qxx
    -0.3788    -0.0647       0.111497102
:
:
:
```

```
:
:
Epoch = 131, Year = 2010, measurement =   0.5946
Filtered State  Corrn         Filtered State cofactor matrix Qxx
     0.6143    -0.0030       0.065887234

Epoch = 132, Year = 2011, measurement =   0.6747
Filtered State  Corrn         Filtered State cofactor matrix Qxx
     0.6222     0.0080       0.065887234

Epoch = 133, Year = 2012, measurement =   1.8681
Filtered State  Corrn         Filtered State cofactor matrix Qxx
     0.7864     0.1642       0.065887234

Epoch = 134, Year = 2013, measurement =   0.2093
Filtered State  Corrn         Filtered State cofactor matrix Qxx
     0.7104    -0.0760       0.065887234

Epoch = 135, Year = 2014, measurement =   0.2816
Filtered State  Corrn         Filtered State cofactor matrix Qxx
     0.6539    -0.0565       0.065887234
```

Figure 15 shows the surface air temperature anomalies (for the 48 contiguous United States) with the Kalman Filter estimates of the anomaly shown as the varying black line. In this case, these estimates are referred to as the signal and it appears to have a general upward trend from 1980 onwards. It may indicate a warming trend for the contiguous 48 United States which accounts for 1.6% of the earth's surface.

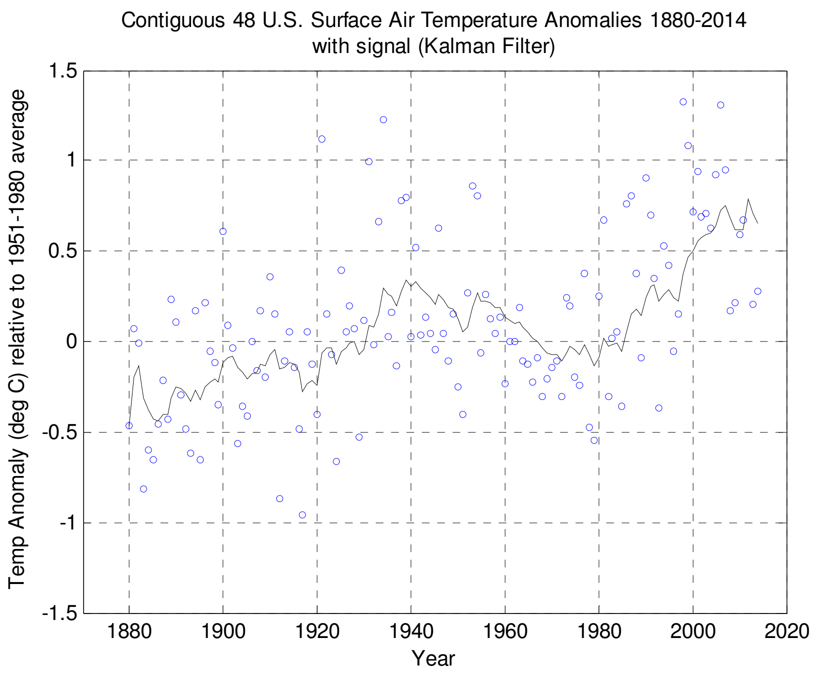


Figure 15. Temperature anomalies with signal from a Kalman Filter

Data from the GISTEMP study is shown below. The data file required for the MATLAB function *global_warming_filter.m* is derived from this data and uses only Year and Annual Mean values.

Contiguous 48 U.S. Surface Air Temperature Anomaly (C)

year	Annual_Mean	5-year_Mean	year	Annual_Mean	5-year_Mean	year	Annual_Mean	5-year_Mean
1880	-0.4656	*	1925	0.3910	-0.0192	1970	-0.1402	-0.2117
1881	0.0693	*	1926	0.0546	0.0092	1971	-0.1071	-0.1025
1882	-0.0067	-0.3650	1927	0.1940	0.0354	1972	-0.3041	-0.0225
1883	-0.8181	-0.4022	1928	0.0689	-0.0194	1973	0.2442	-0.0332
1884	-0.6041	-0.5074	1929	-0.5315	0.1688	1974	0.1945	-0.0598
1885	-0.6516	-0.5487	1930	0.1170	0.1263	1975	-0.1935	0.0758
1886	-0.4563	-0.4715	1931	0.9957	0.2456	1976	-0.2400	-0.0671
1887	-0.2135	-0.3047	1932	-0.0185	0.5966	1977	0.3736	-0.2159
1888	-0.4319	-0.1533	1933	0.6653	0.5794	1978	-0.4702	-0.1265
1889	0.2296	-0.1203	1934	1.2236	0.4127	1979	-0.5493	0.0562
1890	0.1056	-0.1749	1935	0.0308	0.3892	1980	0.2536	-0.0797
1891	-0.2914	-0.2122	1936	0.1621	0.4114	1981	0.6732	0.0184
1892	-0.4865	-0.2249	1937	-0.1360	0.3261	1982	-0.3059	0.1386
1893	-0.6183	-0.3776	1938	0.7767	0.3251	1983	0.0203	0.0167
1894	0.1661	-0.2767	1939	0.7967	0.3963	1984	0.0519	0.0341
1895	-0.6577	-0.1899	1940	0.0260	0.4310	1985	-0.3559	0.2569
1896	0.2127	-0.0892	1941	0.5180	0.3031	1986	0.7599	0.3287
1897	-0.0525	-0.1915	1942	0.0375	0.1530	1987	0.8083	0.3007
1898	-0.1144	0.0626	1943	0.1371	0.1398	1988	0.3794	0.5527
1899	-0.3457	0.0382	1944	0.0466	0.1616	1989	-0.0884	0.5398
1900	0.6131	0.0411	1945	-0.0403	0.1630	1990	0.9044	0.4475
1901	0.0905	-0.0490	1946	0.6273	0.1140	1991	0.6953	0.2979
1902	-0.0379	-0.0509	1947	0.0444	0.1355	1992	0.3468	0.4215
1903	-0.5649	-0.2553	1948	-0.1082	0.0937	1993	-0.3684	0.3251
1904	-0.3552	-0.2727	1949	0.1544	-0.1126	1994	0.5293	0.1762
1905	-0.4088	-0.2966	1950	-0.2492	-0.0686	1995	0.4226	0.1374
1906	0.0035	-0.1496	1951	-0.4044	0.1246	1996	-0.0495	0.4759
1907	-0.1576	-0.1184	1952	0.2645	0.2550	1997	0.1531	0.5874
1908	0.1699	0.0354	1953	0.8578	0.2919	1998	1.3242	0.6458
1909	-0.1991	0.0651	1954	0.8065	0.4248	1999	1.0868	0.8441
1910	0.3602	-0.0774	1955	-0.0647	0.3967	2000	0.7143	0.9509
1911	0.1519	-0.1336	1956	0.2597	0.2343	2001	0.9423	0.8272
1912	-0.8701	-0.0822	1957	0.1243	0.1005	2002	0.6869	0.7348
1913	-0.1109	-0.1831	1958	0.0457	0.0670	2003	0.7059	0.7759
1914	0.0580	-0.3103	1959	0.1376	0.0154	2004	0.6246	0.8488
1915	-0.1444	-0.3281	1960	-0.2321	-0.0103	2005	0.9197	0.9005
1916	-0.4841	-0.2953	1961	0.0013	0.0189	2006	1.3067	0.7931
1917	-0.9593	-0.3317	1962	-0.0040	-0.0295	2007	0.9454	0.7112
1918	0.0534	-0.3834	1963	0.1917	-0.0073	2008	0.1693	0.6462
1919	-0.1241	-0.0635	1964	-0.1044	-0.0528	2009	0.2151	0.5198
1920	-0.4031	0.1584	1965	-0.1213	-0.0694	2010	0.5946	0.7044
1921	1.1156	0.1331	1966	-0.2261	-0.1681	2011	0.6747	0.7124
1922	0.1502	0.0254	1967	-0.0871	-0.1883	2012	1.8681	0.7257
1923	-0.0731	0.1842	1968	-0.3018	-0.1921	2013	0.2093	*
1924	-0.6626	-0.0280	1969	-0.2054	-0.1683	2014	0.2816	*

Anomalies and centred 5-Year Moving Averages for 1880-2014 from
http://data.giss.nasa.gov/gistemp/graphs_v3/Fig.D.txt
Anomalies referenced to absolute global mean temperature for 1951-1980 of 14.0 deg-C

REFERENCES

*Bossler, John D. 1972, 'Bayesian Inference in Geodesy', Ph.D. Dissertation, Department of Geodetic Science, Ohio State University, Columbus, Ohio, USA.

Brown, R.G. and Hwang, P.Y.C, 1992, *Introduction to Random Signals and Applied Kalman Filtering,* 2nd ed., John Wiley & Sons, New York.

*Cross, P.A. 1992, *Advanced Least Squares Applied to Position Fixing,* Working Paper No. 6, Department of Land Information, University of East London. https://seabedhabitats.files.wordpress.com/2011/10/cross_1994.pdf (accessed 10-Aug-2015)

*Gauss, C.F. 1809, *Theory of the Motion of the Heavenly Bodies Moving about the Sun in Conic Sections,* a translation of *Theoria Motus Corporum Coelestium in sectionibus conicis solem ambientium* by C.H. Davis, Dover, New York, 1963.

Hetland, A, 2012, *An Example of Kalman Filtering: The Surface Temperature Anomaly,* Andreas Hetlannds Blog, published 29-Apr-2012, viewed 04-Apr-2015, http://andreashetland.com/blog/2012/04/an-example-of-kalman-filtering-the-surface-air-temperature -anomaly/

*Kalman, R.E. 1960, 'A new approach to linear filtering and prediction', *Journal of Basic Engineering,* American Society of Mechanical Engineers (ASME), Series 82D, pp. 35-45, March 1960.

*Krakiwsky, E.J. 1975, *A Synthesis of Recent Advances in the Method of Least Squares,* Lecture Notes No. 42, 1992 reprint, Department of Surveying Engineering, University of New Brunswick, Fredericton, Canada

*Leahy, F.J. 1974, 'Two hundred years of adjustment of survey measurements', *Two Centuries of Surveying: Proceedings of the 17th Australian Survey Congress,* Melbourne, 23 Feb. – 4 Mar. 1974, Institution of Surveyors, Australia, pp.19-29.

*Merriman, M. 1905, *Method of Least Squares,* 8th edn, John Wiley & Sons, New York.

*Mikhail, E.M. 1976, *Observations and Least Squares,* IEP–A Dun-Donnelley, New York.

*Mikhail, E.M. and Gracie, G., 1981, *Analysis and Adjustment of Survey Measurements,* Van Nostrand Reinhold Company, New York.

*Wells, D.E. and Krakiwsky, E.J. 1971. *The Method of Least Squares.* Lecture Notes No. 18, Department of Surveying Engineering, University of New Brunswick, May 1971, reprinted September 1992, 180 pages.

Sorenson, Harold W. 1985, *Kalman Filtering: Theory and Application,* The Institute of Electrical and Electronics Engineers (IEEE) Press selected reprint series, Editor Harold W. Sorenson, IEEE Press, New York.

APPENDIX: MATLAB FUNCTIONS

MATLAB function *least_squares.m*

```
function least_squares
%
% LEAST_SQUARES reads an ASCII textfile containing data for a least squares
%  adjustment of indirect observations in the matrix form v + Bx = f where
%  v is an n,1 vector of residuals, B is an n,u coefficient matrix, x is a
%  u,1 vector of unknowns and f is an n,1 vector of constant terms.  n is
%  the number of equations and u is the number of unknowns.  The data textfile
%  contains n rows of data with u+2 columns.  The elements of B are
%  contained in the first u columns.  The numeric terms (f) is contained in
%  column u+1 and the weight of each observation is contained in column
%  u+2.  Results are written to a textfile having the same path and name as
%  the data file but with the extension ".out"

%==========================================================================
% Function:  least_squares
%
% Author:
%  Rod Deakin,
%  Department of Geospatial Science, RMIT University,
%  GPO Box 2476V, MELBOURNE VIC 3001
%  AUSTRALIA
%  email: randm.deakin@gmail.com
%
% Date:
%  Version 1.0  14 April  2003
%  Version 1.1   9 June   2003
%  Version 1.1a 12 August 2015 (Change of email address)
%
% Remarks:
%  This function reads numeric data from a textfile containing data for a
%  least squares adjustment of indirect observations in the form of
%  v + Bx = f where v is an n,1 vector of residuals, B is an n,u matrix of
%  coefficients, x in a u,1 vector of unknown parameters and f is an n,1
%  vector of numeric terms.  n is the number of equations and u is the
%  number of unknowns.  The data text file contains n rows of data in u+2
%  columns.  Column u+1 are the numeric terms, column u+2 are the weights
%  of each observation.  Results are written to a textfile having the same
%  path name as the data file but with the extension ".out"
%
% Arrays:
%  B       - (n,u) coeff matrix of observation equation v + Bx = f
%  f       - (n,1) vector of numeric terms
%  N       - (u,u) coefficient matrix of Normal equations Nx = t
%  Ninv    - (u,u) inverse of N
%  Qxx     - (u,u) cofactor matrix of unknowns
%  Qvv     - (n,n) cofactor matrix of residuals
%  Qll     - (n,n) cofactor matrix of adjusted measurements
%  t       - (u,1) vector of numeric terms of Normal equations Nx = t
%  v       - (n,1) vector of residuals
%  W       - (n,n) weight matrix
%  w       - (n,1) vector of weights
%  x       - (u,1) vector of solutions
%
% Variables
%  n       - number of equations
%  u       - number of unknowns
%  vfact   - variance factor
%
```

```
% References:
%  Notes on Least Squares (2003), Department of Geospatial Science, RMIT
%  University, 2003
%
%=============================================================================

%--------------------------------------------------------------------------
% 1. Call the User Interface (UI) to choose the input data file name
% 2. Concatenate strings to give the path and file name of the input file
% 3. Strip off the extension from the file name to give the rootName
% 4. Add extension ".out" to rootName to give the output filename
% 5. Concatenate strings to give the path and file name of the output file
%--------------------------------------------------------------------------
filepath = strcat('d:\temp\','*.txt');
[infilename,inpathname] = uigetfile(filepath);
infilepath = strcat(inpathname,infilename);
rootName   = strtok(infilename,'.');
outfilename = strcat(rootName,'.out');
outfilepath = strcat(inpathname,outfilename);

%----------------------------------------------------------
% 1. Load the data into an array whose name is the rootName
% 2. set fileTemp = rootName
% 3. Copy columns of data into individual arrays
%----------------------------------------------------------
load(infilepath);
fileTemp = eval(rootName);
% get the number of rows (n) and the number of columns (m)
[n,m] = size(fileTemp);
% set the number of unknowns
u = m-2;
% copy the data into B, f and w
B = fileTemp(:,1:u);
f = fileTemp(:,u+1);
w = fileTemp(:,m);

% set the elements of the weight matrix W
W = zeros(n,n);
for k = 1:n
  W(k,k) = w(k);
end

% form the normal equation coefficient matrix N
N = B'*W*B;

% form the vector of numeric terms t
t = B'*W*f;

% solve the system Nx = t for the unknown parameters x
Ninv = inv(N);
x = Ninv*t;

% compute residuals
v = f - (B*x);

% compute the variance factor
vfact = (f'*W*f - x'*t)/(n-u);

% compute the cofactor matrix of the adjusted quantities
Qxx = Ninv;

% compute the cofactor matrix of the residuals
Qvv = inv(W)-B*Ninv*B';

% compute the cofactor matrix of the adjusted quantities
Qll = inv(W)-Qvv;
```

```
% open the output file print the data
fidout  = fopen(outfilepath,'wt');

fprintf(fidout,'\n\nLeast Squares Adjustment of Indirect Observations');
fprintf(fidout,'\n\nInput Data');
fprintf(fidout,'\n\nCoefficient matrix B of observation equations v + Bx = f');
for j = 1:n
  fprintf(fidout,'\n');
  for k = 1:u
    fprintf(fidout,'%15.6f',B(j,k));
  end
end

fprintf(fidout,'\n\nVector of numeric terms f and weights w of observation equations v + Bx =
f');
for k = 1:n
  fprintf(fidout,'\n%15.6f  %15.6f',f(k,1),w(k));
end

fprintf(fidout,'\n\nCoefficient matrix N of Normal equations Nx = t');
fprintf(fidout,'\n(upper triangular part)');
for j = 1:u
  fprintf(fidout,'\n');
  count = 0;
  for k = j:u
    if count > 4
      count = 0;
      fprintf(fidout,'\n');
    end
    fprintf(fidout,'%15.6f',N(j,k));
    count = count + 1;
  end
end

fprintf(fidout,'\n\nVector of numeric terms t of Normal equations Nx = t');
for k = 1:u
  fprintf(fidout,'\n%15.6f',t(k,1));
end

fprintf(fidout,'\n\nInverse of Normal equation coefficient matrix');
fprintf(fidout,'\n(upper triangular part)');
for j = 1:u
  fprintf(fidout,'\n');
  count = 0;
  for k = j:u
    if count > 4
      count = 0;
      fprintf(fidout,'\n');
    end
    fprintf(fidout,'%16.4e',Ninv(j,k));
    count = count + 1;
  end
end

fprintf(fidout,'\n\nVector of solutions x');
for k = 1:u
  fprintf(fidout,'\n%20.9e',x(k,1));
end

fprintf(fidout,'\n\nVector of residuals v');
for k = 1:n
  fprintf(fidout,'\n%12.6f',v(k,1));
end

fprintf(fidout,'\n\nVariance factor = %10.6e',vfact);
fprintf(fidout,'\n\nCofactor matrix of unknowns Qxx');
fprintf(fidout,'\n(upper triangular part)');
for j = 1:u
  fprintf(fidout,'\n');
  count = 0;
  for k = j:u
    if count > 4
      count = 0;
      fprintf(fidout,'\n');
    end
    fprintf(fidout,'%16.4e',Qxx(j,k));
    count = count + 1;
  end
end

fprintf(fidout,'\n\nCofactor matrix of residuals Qvv');
fprintf(fidout,'\n(upper triangular part)');
for j = 1:n
  fprintf(fidout,'\n');
  count = 0;
  for k = j:n
    if count > 4
      count = 0;
      fprintf(fidout,'\n');
    end
    fprintf(fidout,'%16.4e',Qvv(j,k));
    count = count + 1;
  end
end

fprintf(fidout,'\n\nCofactor matrix of adjusted observations Qll');
fprintf(fidout,'\n(upper triangular part)');
for j = 1:n
  fprintf(fidout,'\n');
  count = 0;
  for k = j:n
    if count > 4
      count = 0;
      fprintf(fidout,'\n');
    end
    fprintf(fidout,'%16.4e',Qll(j,k));
    count = count + 1;
  end
end

fprintf(fidout,'\n\n');

% close the output file
fclose(fidout);
```

Help message for MATLAB function *least_squares.m*

```
>> help least_squares
  least_squares reads an ASCII textfile containing data for a least squares
   adjustment of indirect observations in the matrix form v + Bx = f where
   v is an n,1 vector of residuals, B is an n,u coefficient matrix, x is a
   u,1 vector of unknowns and f is an n,1 vector of constant terms.  n is
   the number of equations and u is the number of unknowns.  The data textfile
   contains n rows of data with u+2 columns.  The elements of B are
   contained in the first u columns.  The numeric terms (f) is contained in
   column u+1 and the weight of each observation is contained in column
   u+2.  Results are written to a textfile having the same path and name as
   the data file but with the extension ".out"

>>
```

Data file (`Example_1.txt`) for MATLAB function *least_squares.m*

```
% Data file for function "least_squares.m"
% Example 1: best fit line
%  B(1)   B(2)   f     w
   40.0   -1    24.0   2
   15.0   -1    24.0   5
  -10.0   -1    12.0   7
  -38.0   -1   -15.0   3
  -67.0   -1   -30.0   3
```

Output file (`Example_1.out`) from MATLAB function *least_squares.m*

```
Least Squares Adjustment of Indirect Observations

Input Data

Coefficient matrix B of observation equations v + Bx = f
      40.000000      -1.000000
      15.000000      -1.000000
     -10.000000      -1.000000
     -38.000000      -1.000000
     -67.000000      -1.000000

Vector of numeric terms f and weights w of observation equations v + Bx = f
      24.000000         2.000000
      24.000000         5.000000
      12.000000         7.000000
     -15.000000         3.000000
     -30.000000         3.000000

Coefficient matrix N of Normal equations Nx = t
(upper triangular part)
   22824.000000     230.000000
      20.000000

Vector of numeric terms t of Normal equations Nx = t
   10620.000000
    -117.000000

Inverse of Normal equation coefficient matrix
(upper triangular part)
      4.9556e-05     -5.6990e-04
      5.6554e-02
Vector of solutions x
     5.929679370e-01
    -1.266913128e+01
```

```
Vector of residuals v
  -12.387849
    2.436350
    5.260548
   -5.136350
   -2.940279

Variance factor = 2.117974e+02

Cofactor matrix of unknowns Qxx
(upper triangular part)
      4.9556e-05     -5.6990e-04
      5.6554e-02

Cofactor matrix of residuals Qvv
(upper triangular part)
      3.1856e-01     -1.1763e-01     -5.3828e-02      1.7632e-02      9.1645e-02
      1.1520e-01     -5.1970e-02     -1.5199e-02      2.2885e-02
      9.2746e-02     -4.8030e-02     -4.5874e-02
      2.4853e-01     -1.2289e-01
      1.3069e-01

Cofactor matrix of adjusted observations Qll
(upper triangular part)
      1.8144e-01      1.1763e-01      5.3828e-02     -1.7632e-02     -9.1645e-02
      8.4801e-02      5.1970e-02      1.5199e-02     -2.2885e-02
      5.0112e-02      4.8030e-02      4.5874e-02
      8.4801e-02      1.2289e-01
      2.0265e-01

Cofactor matrix of residuals Qvv
(upper triangular part)
      4.2118e-01     -3.9669e-01     -2.1457e-01     -1.0591e-02      2.0067e-01
      6.9787e-01     -2.0757e-01     -1.0165e-01      8.0415e-03
      7.9944e-01     -1.9272e-01     -1.8459e-01
      7.0530e-01     -4.0034e-01
      3.7621e-01

Cofactor matrix of adjusted observations Qll
(upper triangular part)
      5.7882e-01      3.9669e-01      2.1457e-01      1.0591e-02     -2.0067e-01
      3.0213e-01      2.0757e-01      1.0165e-01     -8.0415e-03
      2.0056e-01      1.9272e-01      1.8459e-01
      2.9470e-01      4.0034e-01
      6.2379e-01
```

MATLAB function *linear_regression_CLS.m*

```
function linear_regression_CLS
%
% linear_regression_CLS is a function that fits a straight line y = b*x + c
%   to a set of x,y data points using Combined Least Squares that allows
%   for proper consideration of variances and covariances of the x,y data.
%   The function reads an ASCII textfile containing coordinate pairs (x,y)
%   standard deviations (sx,sy) and covariance (sxy).  In addition point
%   name is included in the textfile.
%   Part of an ASCII textfile is shown below
%
% Data for Example 2
%
% Serial    x       sd_x           y      sd_y        covariance
%   1      -40.0    1.414213562   -24.0   1.732050808     0.5
%   2      -15.0    2.828427125   -24.0   2.236067978    -4.0
%   3       10.0    2.828427125   -12.0   2.645751311    -3.0
%   4       38.0    1.0            15.0   1.414213562     0.5
%   5       67.0    2.449489743    30.0   3.464101615     1.0
%
%   Results are written to a textfile having the same path and name as the
%   data file but with the extension ".out"

%============================================================================
% Function:  linear_regression_CLS
%
% Author:
%  Rod Deakin,
%  BONBEACH VIC 3196
%  AUSTRALIA
%  email: randm.deakin@gmail.com
%
% Date:
%  Version 1.0  10 November 2014
%  Version 1.1  12 August   2015
%
% Remarks:
%  This function reads numeric data from a textfile containing coordinate
%  pairs (x,y) standard deviation and covariances associated with each pair
%  and computes the parameters m and c of a line of best fit y = b*x + c
%  using Combined Least Squares.  This allows proper consideration of the
%  standard deviations and covariances of the measurements (x,y coords).
%  Results are written to the screen and to textfile having the same path
%  and name as the data file but with the extension ".out"
%
% Arrays:
%  A       - (m,n) coeff matrix of observation equations A*v + B*x = f
%  B       - (m,u) coeff matrix of observation equations A*v + B*x = f
%  f       - (m,1) vector of numeric terms
%  k       - (m,1) vector of Lagrange multipliers k = We*(f-B*x)
%  N       - (u,u) coefficient matrix of Normal equations N*x = t and
%                  N = Bt*We*B
%  Ninv    - (u,u) inverse of coefficient matrix of Normal equations
%  Q       - (n,n) matrix of estimates of variances and covariances
%  Qll     - (n,n) cofactor matrix of adjusted observations
%  Qvv     - (n,n) cofactor matrix of residuals
%  We      - (m,m) equivalent weight matrix We = inv(A*Q*At)
%  x_coord - (m,1) vector of x coordinates
%  y_coord - (m,1) vector of y coordinates
%  serial  - (m,1) vector of station serial numbers
%  sx      - (m,1) vector of st. dev's of x coordinates
%  sxy     - (m,1) vector of covariances between x and y coordinates
%  sy      - (m,1) vector of st. dev's of y coordinates
%  t       - (u,1) vector of numeric terms of Normal equations N*x = t and
%                  t = Bt*We*f
%  v       - (n,1) vector of residuals v = Q*At*k
%  W       - (n,n) matrix of weight coefficients W = inv(Q)
%  x       - (u,1) vector of solutions x = Ninv*t
%
% Variables
%  b       - b = tan(theta) = (y2-y1)/(x2-x1) = slope of straight line
%  b0      - approximate value of slope of straight line
%  c       - point on y-axis where straight line intersects
%  db      - small correction to slope of line b = b0 + db
%  eps     - small quantity for testing for convergence
%  iter    - iteration count
%  m       - number of observation equations equal to number of (x,y) pairs
%  n       - number of observations (or measurements)
%  u       - number of unknowns (parameters m and c)
%  i,j,kk  - integer counters
%  vfact   - variance factor
%  xc,yc   - x and y centroids (averages)
%  x1,x2   - x coords of first and last data points
%  y1,y2   - y coords of first and last data points
%
% References:
%  Deakin, R.E., 2014, 'Linear Regression Using Combined Least Squares',
%     Private Notes, Bonbeach VIC 3196, Version 1, 03-Nov-2014
%
%============================================================================

%--------------------------------------------------------------------------
% 1. Call the User Interface (UI) to choose the input data file name
% 2. Concatenate strings to give the path and file name of the input file
% 3. Strip off the extension from the file name to give the rootName
% 4. Add extension ".out" to rootName to give the output filename
% 5. Concatenate strings to give the path and file name of the output file
%--------------------------------------------------------------------------
filepath = strcat('d:\temp\','*.txt');
[infilename,inpathname] = uigetfile(filepath);
infilepath  = strcat(inpathname,infilename);
rootName    = strtok(infilename,'.');
outfilename = strcat(rootName,'.out');
outfilepath = strcat(inpathname,outfilename);

%----------------------------------------------------------
% 1. Load the data into an array whose name is the rootName
% 2. set fileTemp = rootName
% 3. Copy columns of data into individual arrays
%----------------------------------------------------------
load(infilepath)
fileTemp = eval(rootName);
serial  = fileTemp(:,1);
x_coord = fileTemp(:,2);
sx      = fileTemp(:,3);
y_coord = fileTemp(:,4);
sy      = fileTemp(:,5);
sxy     = fileTemp(:,6);
%---------------------------------------------------------------------------
% Determine the adjustment constants m, n and u
%---------------------------------------------------------------------------
% determine the number of observation equations
m = length(x_coord);
% set the number of measurements
n = 2*m;
% set the number of unknowns
u = 2;
```

```
%--------------------------------------------------------------------------
% Print heading and given data
%--------------------------------------------------------------------------
% open the output file print the data
fidout  = fopen(outfilepath,'wt');
fprintf(fidout,'\n\n Linear Regression Using Combined Least Squares');
fprintf(fidout,'\n =============================================');
fprintf(fidout,'\n solution for the parameters b and c in the equation for the line of best
fit');
fprintf(fidout,'\n y = b*x + c  where b = tan(theta) is the slope and c is the intercept on the
y-axis');
fprintf(fidout,'\n\n Input Data');
fprintf(fidout,'\n Serial      x       sx              y       sy            sxy');
for kk = 1:m
    fprintf(fidout,'\n %4d %12.4f  %8.6f  %12.4f  %8.6f     %9.6f',...
            serial(kk,1),x_coord(kk,1),sx(kk,1),y_coord(kk,1),...
            sy(kk,1),sxy(kk,1));
end
fprintf(fidout,'\n\n there are n = %2d measurements (the x,y coordinates),',n);
fprintf(fidout,'\n m = %2d observation equations and u = %2d unknowns in the',m,u);
fprintf(fidout,'\n system of equations A*v + B*x = f');
%--------------------------------------------------------------------------
% set the elements of the cofactor matrix of observations Q
%--------------------------------------------------------------------------
Q = zeros(n,n);
for kk = 1:m
    j = 2*kk;
    Q(j-1,j-1) = sx(kk,1)^2;  % variance x = (sx)^2
    Q(j-1,j) = sxy(kk,1);     % covariance
    Q(j,j-1) = Q(j-1,j);      % covariance
    Q(j,j) = sy(kk,1)^2;      % variance y = (sy)^2
end
%--------------------------------------------------------------------------
% Reduce the coordinates to a centroidal system
%--------------------------------------------------------------------------
xc = sum(x_coord)/m;
yc = sum(y_coord)/m;
x_coord = x_coord-xc;
y_coord = y_coord-yc;
%--------------------------------------------------------------------------
% form the coefficient matrix B of the observation equation A*v +B*x = f
%--------------------------------------------------------------------------
B = zeros(m,u);
for kk = 1:m
    B(kk,1) = -x_coord(kk,1);
    B(kk,2) = -1.0;
end
%--------------------------------------------------------------------------
% set approximate value for slope b0 where b = b0 + db and db is a small
% correction.  b = tan(theta) = (y2-y1)/(x2-x1)
%--------------------------------------------------------------------------
y1 = y_coord(1,1);
y2 = y_coord(m,1);
x1 = x_coord(1,1);
x2 = x_coord(m,1);
b0 = (y2-y1)/(x2-x1);

%--------------------------------------------------------------------------
% Solution of least squares estimates by iteration
%--------------------------------------------------------------------------
% while loop for the computation of solution vector x where x(1,1) = db
% and x(2,1) = c.  Break from while loop when abs(db) < eps
% set the smmal increment to the slope where b = b0 + db
db = 10;
% set the iteration number
iter = 1;
```

```
% set eps, a small quantity
eps = 1.0e-8;
while 1
    iter;
    if abs(db)<eps
        break
    end
    if iter > 10
        break
    end
    % form the coefficient matrix A of observation equations A*v + B*x = f
    A = zeros(m,n);
    for kk = 1:m
        j = 2*kk;
        A(kk,j)   = 1.0;
        A(kk,j-1) = -b0;
    end
    % form the vector of numeric terms f
    f = zeros(m,1);
    for kk = 1:m
        f(kk,1) = b0*x_coord(kk,1)-y_coord(kk,1);
    end
    % form ther equivalent weight matrix We
    We = inv(A*Q*A');
    % form the normal equation coefficient matrix N
    N = B'*We*B;
    % form the vector of numeric terms t
    t = B'*We*f;
    % solve the system Nx = t for the unknown parameters x
    Ninv = inv(N);
    x = Ninv*t;

    if iter == 1
        % print matrices A, B, f, Q, We, N, t and x
        fprintf(fidout,'\n\n System of equations and solution for first iteration');
        fprintf(fidout,'\n where approximate slope b0 = % 12.9f',b0);
        fprintf(fidout,'\n Coordinates have been reduced to a centroidal system');
        fprintf(fidout,'\n The * symbol indicates new row of a matrix');
        fprintf(fidout,'\n Coefficient matrix A');
        for i = 1:m
            kk = 1;
            fprintf(fidout,'\n* ');
            for j = 1:n
                if (kk > 6)
                    fprintf(fidout,'\n  ');
                    kk = 1;
                end
                fprintf(fidout,'% 12.9f ',A(i,j));
                kk = kk+1;
            end
        end
        fprintf(fidout,'\n Coefficient matrix B');
        for i = 1:m
            kk = 1;
            fprintf(fidout,'\n* ');
            for j = 1:u
                if (kk > 6)
                    fprintf(fidout,'\n  ');
                    kk = 1;
                end
                fprintf(fidout,'% 12.9f ',B(i,j));
                kk = kk+1;
            end
        end
        fprintf(fidout,'\n Vector f-transposed');
        kk = 1;
        fprintf(fidout,'\n  ');
```

```
for i = 1:m
    if (kk > 6)
        fprintf(fidout,'\n  ');
        kk = 1;
    end
    fprintf(fidout,'% 12.9f ',f(i,1));
    kk = kk+1;
end
fprintf(fidout,'\n Cofactor matrix Q');
for i = 1:n
    kk = 1;
    fprintf(fidout,'\n* ');
    for j = 1:n
        if (kk > 6)
            fprintf(fidout,'\n  ');
            kk = 1;
        end
        fprintf(fidout,'% 12.9f ',Q(i,j));
        kk = kk+1;
    end
end
fprintf(fidout,'\n Equivalent weight matrix We = inv(A*Q*At)');
for i = 1:m
    kk = 1;
    fprintf(fidout,'\n* ');
    for j = 1:m
        if (kk > 6)
            fprintf(fidout,'\n  ');
            kk = 1;
        end
        fprintf(fidout,'% 15.9f ',We(i,j));
        kk = kk+1;
    end
end
fprintf(fidout,'\n Matrix N = Bt*We*B');
for i = 1:u
    kk = 1;
    fprintf(fidout,'\n* ');
    for j = 1:u
        if (kk > 6)
            fprintf(fidout,'\n  ');
            kk = 1;
        end
        fprintf(fidout,'% 18.9f ',N(i,j));
        kk = kk+1;
    end
end
fprintf(fidout,'\n Vector t-transposed where t = Bt*We*f');
kk = 1;
fprintf(fidout,'\n  ');
for i = 1:u
    if (kk > 6)
        fprintf(fidout,'\n  ');
        kk = 1;
    end
    fprintf(fidout,'% 12.9f ',t(i,1));
    kk = kk+1;
end
fprintf(fidout,'\n Vector of solutions x-transposed where x = Ninv*t');
kk = 1;
fprintf(fidout,'\n  ');
for i = 1:u
    if (kk > 6)
        fprintf(fidout,'\n  ');
        kk = 1;
    end
    fprintf(fidout,'% 12.9f ',x(i,1));
```

```
            kk = kk+1;
        end
    end
    % update value of b0
    db = x(1,1);
    b0 = b0+db;
    % increment iteration count
    iter = iter+1;
end
%--------------------------------------------------------------------------
% Set the solutions for slope b and y-axis intercept c for the straight
% line y = bx + c
%--------------------------------------------------------------------------
db = x(1,1);
b  = b0+db;
c  = b*(-xc)+x(2,1)+yc;
%--------------------------------------------------------------------------
% compute vector of Lagrange multipliers k and vector of residuals v
%--------------------------------------------------------------------------
k = We*(f-B*x);
v = Q*A'*k;

%--------------------------------------------------------------------------
% compute the variance factor and cofactor matrices Qxx, Qll, Qvv
%--------------------------------------------------------------------------
W = inv(Q);
vfact = (v'*W*v)/(m-u);

% compute the cofactor matrix of the adjusted quantities
Qxx = Ninv;

% compute the cofactor matrix of the adjusted quantities
Qll = Q + Q*A'*We*B*Ninv*B'*We*A*Q - Q*A'*We*A*Q;

% compute the cofactor matrix of the residuals
Qvv = Q - Qll;

%--------------------------------------------------------------------------
% Print solutions
%--------------------------------------------------------------------------
fprintf(fidout,'\n\n Solutions after %2d iterations',iter);
fprintf(fidout,'\n ==============================');
fprintf(fidout,'\n\n slope of line m = tan(theta) = % 8.6f',b');
fprintf(fidout,'\n y-axis intercept           c = % 8.6f',c');
fprintf(fidout,'\n\n Vector of Lagrange multipliers k-transposed where k = We*(f-B*x)');
kk = 1;
fprintf(fidout,'\n  ');
for i = 1:m
    if (kk > 6)
        fprintf(fidout,'\n  ');
        kk = 1;
    end
    fprintf(fidout,'% 12.9f ',k(i,1));
    kk = kk+1;
end
fprintf(fidout,'\n\n Vector of residuals v-transposed where v = Q*At*k');
kk = 1;
fprintf(fidout,'\n  ');
for i = 1:n
    if (kk > 6)
        fprintf(fidout,'\n  ');
        kk = 1;
    end
    fprintf(fidout,'% 12.9f ',v(i,1));
    kk = kk+1;
end
```

```
fprintf(fidout,'\n\n Variance factor = %10.6e',vfact);

fprintf(fidout,'\n\n Cofactor matrix of unknowns Qxx (upper triangular part)');
fprintf(fidout,'\n The * symbol indicates new row beginning at diagonal');
for j = 1:u
  fprintf(fidout,'\n* ');
  count = 0;
  for kk = j:u
    if count > 5
      count = 0;
      fprintf(fidout,'\n  ');
    end
    fprintf(fidout,'%14.6e',Qxx(j,kk));
    count = count + 1;
  end
end

fprintf(fidout,'\n\n Cofactor matrix of residuals Qvv (upper triangular part)');
fprintf(fidout,'\n The * symbol indicates new row beginning at diagonal');
for j = 1:n
  fprintf(fidout,'\n* ');
  count = 0;
  for kk = j:n
    if count > 5
      count = 0;
      fprintf(fidout,'\n  ');
    end
    fprintf(fidout,'%14.6e',Qvv(j,kk));
    count = count + 1;
  end
end

fprintf(fidout,'\n\n Cofactor matrix of adjusted observations Qll (upper triangular part)');
fprintf(fidout,'\n The * symbol indicates new row beginning at diagonal');
for j = 1:n
  fprintf(fidout,'\n* ');
  count = 0;
  for kk = j:n
    if count > 5
      count = 0;
      fprintf(fidout,'\n  ');
    end
    fprintf(fidout,'%14.6e',Qll(j,kk));
    count = count + 1;
  end
end

fprintf(fidout,'\n\n');

% close the output file
fclose(fidout);
```

Help message for MATLAB function *linear_regression_CLS.m*

```
>> help linear_regression_CLS
  linear_regression_CLS is a function that fits a straight line y = b*x + c
    to a set of x,y data points using Combined Least Squares that allows
    for proper consideration of variances and covariances of the x,y data.
    The function reads an ASCII textfile containing coordinate pairs (x,y)
    standard deviations (sx,sy) and covariance (sxy).  In addition point
    name is included in the textfile.
    Part of an ASCII textfile is shown below

  Data for Example 2

  Serial    x       sd_x           y      sd_y        covariance
    1      -40.0    1.414213562   -24.0   1.732050808     0.5
    2      -15.0    2.828427125   -24.0   2.236067978    -4.0
    3       10.0    2.828427125   -12.0   2.645751311    -3.0
    4       38.0    1.0            15.0   1.414213562     0.5
    5       67.0    2.449489743    30.0   3.464101615     1.0

    Results are written to a textfile having the same path and name as the
    data file but with the extension ".out"

>>
```

Data file (`Example_2.txt`) for MATLAB function *linear_regression_CLS.m*

```
% Data for Example 2
%
%Serial    x       sd_x           y      sd_y        covariance
  1      -40.0    1.414213562   -24.0   1.732050808     0.5
  2      -15.0    2.828427125   -24.0   2.236067978    -4.0
  3       10.0    2.828427125   -12.0   2.645751311    -3.0
  4       38.0    1.0            15.0   1.414213562     0.5
  5       67.0    2.449489743    30.0   3.464101615     1.0
```

Output file (`Example_2.out`) from MATLAB function *linear_regression_CLS.m*

```
Linear Regression Using Combined Least Squares
 ==============================================
 solution for the parameters b and c in the equation for the line of best fit
 y = b*x + c  where b = tan(theta) is the slope and c is the intercept on the y-axis

 Input Data
 Serial      x       sx             y       sy            sxy
    1     -40.0000  1.414214      -24.0000  1.732051      0.500000
    2     -15.0000  2.828427      -24.0000  2.236068     -4.000000
    3      10.0000  2.828427      -12.0000  2.645751     -3.000000
    4      38.0000  1.000000       15.0000  1.414214      0.500000
    5      67.0000  2.449490       30.0000  3.464102      1.000000

 there are n = 10 measurements (the x,y coordinates),
 m =  5 observation equations and u =  2 unknowns in the
 system of equations A*v + B*x = f

 System of equations and solution for first iteration
 where approximate slope b0 =  0.504672897
 Coordinates have been reduced to a centroidal system
 The * symbol indicates new row of a matrix
 Coefficient matrix A
* -0.504672897  1.000000000  0.000000000  0.000000000  0.000000000  0.000000000
   0.000000000  0.000000000  0.000000000  0.000000000
*  0.000000000  0.000000000 -0.504672897  1.000000000  0.000000000  0.000000000
   0.000000000  0.000000000  0.000000000  0.000000000
*  0.000000000  0.000000000  0.000000000  0.000000000 -0.504672897  1.000000000
```

```
   0.000000000  0.000000000  0.000000000  0.000000000
*  0.000000000  0.000000000  0.000000000  0.000000000  0.000000000  0.000000000
  -0.504672897  1.000000000  0.000000000  0.000000000
*  0.000000000  0.000000000  0.000000000  0.000000000  0.000000000  0.000000000
   0.000000000  0.000000000 -0.504672897  1.000000000
 Coefficient matrix B
*  52.000000000 -1.000000000
*  27.000000000 -1.000000000
*  2.000000000 -1.000000000
* -26.000000000 -1.000000000
* -55.000000000 -1.000000000
 Vector f-transposed
  -5.242990654  7.373831776  7.990654206 -4.878504673 -5.242990654
 Cofactor matrix Q
*  1.999999999  0.500000000  0.000000000  0.000000000  0.000000000  0.000000000
   0.000000000  0.000000000  0.000000000  0.000000000
*  0.500000000  3.000000001  0.000000000  0.000000000  0.000000000  0.000000000
   0.000000000  0.000000000  0.000000000  0.000000000
*  0.000000000  0.000000000  8.000000001 -4.000000000  0.000000000  0.000000000
   0.000000000  0.000000000  0.000000000  0.000000000
*  0.000000000  0.000000000 -4.000000000  5.000000002  0.000000000  0.000000000
   0.000000000  0.000000000  0.000000000  0.000000000
*  0.000000000  0.000000000  0.000000000  0.000000000  8.000000001 -3.000000000
   0.000000000  0.000000000  0.000000000  0.000000000
*  0.000000000  0.000000000  0.000000000  0.000000000 -3.000000000  7.000000000
   0.000000000  0.000000000  0.000000000  0.000000000
*  0.000000000  0.000000000  0.000000000  0.000000000  0.000000000  0.000000000
   1.000000000  0.500000000  0.000000000  0.000000000
*  0.000000000  0.000000000  0.000000000  0.000000000  0.000000000  0.000000000
   0.500000000  1.999999999  0.000000000  0.000000000
*  0.000000000  0.000000000  0.000000000  0.000000000  0.000000000  0.000000000
   0.000000000  0.000000000  6.000000001  1.000000000
*  0.000000000  0.000000000  0.000000000  0.000000000  0.000000000  0.000000000
   0.000000000  0.000000000  1.000000000  11.999999999
 Equivalent weight matrix We = inv(A*Q*At)
*     0.332810093     0.000000000     0.000000000     0.000000000     0.000000000
*     0.000000000     0.090293934     0.000000000     0.000000000     0.000000000
*     0.000000000     0.000000000     0.082880287     0.000000000     0.000000000
*     0.000000000     0.000000000     0.000000000     0.571421442     0.000000000
*     0.000000000     0.000000000     0.000000000     0.000000000     0.079879716
 Matrix N = Bt*We*B
*     1593.991326596       -0.659479727
*       -0.659479727        1.157285472
 Vector t-transposed where t = Bt*We*f
   24.079826288  3.623330987
 Vector of solutions x-transposed where x = Ninv*t
   0.016405829  3.140236688

 Solutions after  6 iterations
 ==============================

 slope of line m = tan(theta) =  0.520869
 y-axis intercept           c = -6.082465

 Vector of Lagrange multipliers k-transposed where k = We*(f-B*x)
  -0.965411704  0.891254776  0.904891143 -0.736642565 -0.094091649

 Vector of residuals v-transposed where v = Q*At*k
   0.523000101 -2.644808626 -7.278834590  6.313181624 -6.485310996  7.748227087
   0.015372954 -1.281438011  0.199964859 -1.080090370

 Variance factor = 7.650438e+00

 Cofactor matrix of unknowns Qxx (upper triangular part)
 The * symbol indicates new row beginning at diagonal
*   6.306103e-04  2.900165e-04
*   8.687864e-01
```

```
 Cofactor matrix of residuals Qvv (upper triangular part)
 The * symbol indicates new row beginning at diagonal
*   1.536180e-02 -7.768455e-02 -2.235833e-01  1.939214e-01 -9.600483e-02  1.147003e-01
   -1.851160e-05  1.543065e-03  2.827087e-02 -1.527023e-01
*   3.928503e-01  1.130660e+00 -9.806597e-01  4.854959e-01 -5.800389e-01  9.361308e-05
   -7.803273e-03 -1.429657e-01  7.722148e-01
*   5.201861e+00 -4.511752e+00 -3.755569e-01  4.486910e-01 -3.656917e-03  3.048284e-01
    7.243307e-03 -3.912400e-02
*   3.913196e+00  3.257334e-01 -3.891650e-01  3.171769e-03 -2.643881e-01 -6.282367e-03
    3.393358e-02
*   3.881872e+00 -4.637808e+00 -5.857898e-03  4.882948e-01 -8.019376e-02  4.331586e-01
*   5.540951e+00  6.998635e-03 -5.833828e-01  9.581028e-02 -5.175097e-01
*   6.257971e-05 -5.216435e-03 -3.611610e-03  1.950775e-02
*   4.348246e-01  3.010517e-01 -1.626101e+00
*   2.787822e-01 -1.505814e+00
*   8.133506e+00

 Cofactor matrix of adjusted observations Qll (upper triangular part)
 The * symbol indicates new row beginning at diagonal
*   1.984638e+00  5.776846e-01  2.235833e-01 -1.939214e-01  9.600483e-02 -1.147003e-01
    1.851160e-05 -1.543065e-03 -2.827087e-02  1.527023e-01
*   2.607150e+00 -1.130660e+00  9.806597e-01 -4.854959e-01  5.800389e-01 -9.361308e-05
    7.803273e-03  1.429657e-01 -7.722148e-01
*   2.798139e+00  5.117516e-01  3.755569e-01 -4.486910e-01  3.656917e-03 -3.048284e-01
   -7.243307e-03  3.912400e-02
*   1.086804e+00 -3.257334e-01  3.891650e-01 -3.171769e-03  2.643881e-01  6.282367e-03
   -3.393358e-02
*   4.118128e+00  1.637808e+00  5.857898e-03 -4.882948e-01  8.019376e-02 -4.331586e-01
*   1.459049e+00 -6.998635e-03  5.833828e-01 -9.581028e-02  5.175097e-01
*   9.999374e-01  5.052164e-01  3.611610e-03 -1.950775e-02
*   1.565175e+00 -3.010517e-01  1.626101e+00
*   5.721218e+00  2.505814e+00
*   3.866494e+00
```

MATLAB function *edm.m*

```
function edm
%
% function to simulate the processing of EDM measurements through
% a Kalman filter.

%==========================================================================
% Function:  edm
%
% Author:
%  Rod Deakin,
%  BONBEACH, VIC 3196,
%  AUSTRALIA
%  email: randm.deakin@gmail.com
%
% Date:
%  Version 1.0  15 April  2006
%  Version 1.1  29 August 2015
%
% Remarks:
%  This function simulates the processing of EDM measurements
%  through a Kalman filter.
%
% References:
%  [1] Deakin, R.E., 2015, Least Squares and Kalman Filtering,
%        Lecture Notes, September 2015
%  [2] Deakin, R.E., 2006, The Kalman Filter and Surveying Applications,
%        Presented at the Victorian Regional Surveying Conference,
%        Mildura, 23-25 June 2006.
%
% Arrays:
%  B         - Design matrix, dimensions (m,n)
%  corrn     - array of corrections to State vector, dimensions (n,epochs)
%  H         - coefficient matrix of System Driving Noise, dimensions (n,u)
%  I         - Identity matrix, dimensions (n,n)
%  K         - Gain matrix, dimensions (n,m)
%  meas      - vector of measurements, dimensions (epochs,1)
%  noise     - vector of normally distributed measurement "noise" with
%              a mean of zero and a standard deviation of sigma (epochs,1)
%  Q         - cofactor matrix of measurements, dimensions (n,n)
%  Qmm       - cofactor matrix of Secondary Model, dimensions (n,n)
%  Qww       - cofactor matrix of System Driving Noise, dimensions (n,n)
%  Qxx       - cofactor matrix of State vector, dimensions (n,n)
%  std_xhat  - vector of standard deviations of the filtered State
%  T         - Transition matrix, dimensions (n,n)
%  U         - cofactor update matrix, dimensions (n,n)
%  xhat      - array containing the State vector at each epoch,
%              dimensions (n,epochs)
%
% Variables:
%  const     - a constant value
%  epochs    - number of epochs
%  i,j,k     - integer counters
%  m         - number of measurements at particular epoch
%  n         - number of parameters in the State vector 'xhat'
%  u         - number of parameters in the System Driving Noise vector
%  std_noise - st dev of measurement noise
%==========================================================================
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% define the Kalman filter matrices: %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n = 1;  % number of elements in State vector
m = 1;  % number of measurements at each epoch
u = 1;  % number of elements in System Driving Noise vector
% Set the total number of measurement epochs
epochs = 250;
% Initialize the State vector.
xhat = zeros(n,epochs);
% Initialize the corrections to State vector.
corrn = zeros(n,epochs);
% Set the State Transition matrix T
T = eye(n);
T(1,1) = 1.0;
% Set the cofactor matrix of the System Driving Noise Qww
Qww = zeros(u,u);
Qww(1,1) = 0.0;
% Set the coefficient matrix of System Driving Noise H
H = zeros(n,u);
H(1,1) = 1.0;
% Compute cofactor matrix of Dynamic Model by propagation of variances
Qmm = H*Qww*H';

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Generate some measurements with noise %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Set the state of the random number generator (rng).
% Use rng('default') to ensure that you get the same result as Ref [1].
% Use rng('shuffle') for other results
rng('default');
% rng('shuffle');
% Set the st. dev. of measurement noise, initialise the measurement vector
% then generate the noise
std_noise = 0.010;
meas      = zeros(epochs,1);
noise     = std_noise .* randn(epochs,1);
% Generate the measurement vector
const = 355.420;
for k = 1:epochs
  meas(k) = const + noise(k);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Set Cofactor matrices and State vector to initial values %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Set the cofactor matrix of the measurements Q
Q = zeros(m,m);
Q(1,1)  = std_noise^2;
% Set the State cofactor matrix Qxx.  This will be the filtered
% State cofactor matrix at epoch t2.
Qxx = zeros(n,n);
Qxx(1,1) = std_noise^2;
% Set the starting estimate of the State vector for Epoch 1
% This will be the filtered Sate at epoch t2
xhat(1,1) = meas(1,1);
% Initialize the Gain matrix
K = zeros(n,m);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Start the Kalman Filter at epoch t2 %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for k = 2:epochs
  % Compute the predicted State vector.
  xhat(:,k) = T*xhat(:,k-1);
  % Compute the predicted State cofactor matrix
  Qxx = T*Qxx*T' + Qmm;
  % Set the elements of the design matrix B
  B = zeros(m,n);
  B(1,1) = -1;
  % Compute the Gain matrix K
  K = Qxx*B'/(Q + B*Qxx*B');
  % Compute corrections to predicted State
  corrn(:,k) = K*(-meas(k) - B*xhat(:,k));
  % Compute the filtered State vector
  xhat(:,k) = xhat(:,k) + corrn(:,k);
  % Compute the cofactor update matrix
  I = eye(n);
  U = I - K*B;
  % Compute the filtered State cofactor matrix
  Qxx = U*Qxx;
  std_xhat(k) = sqrt(Qxx(1,1));
  % Print the State vector, corrections and filtered State cofactor matrix
  fprintf('\n\nEpoch = %3d, measurement = %8.4f',k,meas(k));
  fprintf('\nFiltered State  Corrn         Filtered State cofactor matrix Qxx');
  for i=1:u
      fprintf('\n  %9.4f %10.4f      ',xhat(i,k),corrn(i,k));
      for j=1:u
          fprintf('%12.9f',Qxx(i,j));
      end
  end
end
fprintf('\n\n');
%%%%%%%%%%%%%%%%%%%%%%%%%
% End of Kalman filter %%
%%%%%%%%%%%%%%%%%%%%%%%%%

% Figure 1: create a figure window with two plots
figure(1);
clf;
subplot(2,1,1);
hold on;
grid on;
box on;
plot(meas,'k.');
plot(xhat,'k');
title('Kalman Filter estimate');
ylabel('state [m]');

subplot(2,1,2);
hold on;
grid on;
box on;
plot(std_xhat,'k');
title('Standard deviation of estimate');
xlabel('epoch [s]');
ylabel('std [m]');

% Figure 2: Plot of Kalman Filter estimate
figure(2);
clf;
hold on;
grid on;
plot(meas,'k.');
plot(xhat,'k');
hold off;
```

```
title('Kalman Filter estimate');
xlabel('epoch [s]');
ylabel('state [m]');

% Figure 3: Plot of st. dev. of Kalman Filter estimate
figure(3);
clf;
hold on;
grid on;
plot(std_xhat,'k');
hold off;
title('Standard deviation of estimate');
xlabel('epoch [s]');
ylabel('std [m]');
```

Output from MATLAB function *edm.m*

```
>> edm


Epoch =   2, measurement = 355.4383
Filtered State  Corrn         Filtered State cofactor matrix Qxx
   355.4319     0.0065       0.000050000

Epoch =   3, measurement = 355.3974
Filtered State  Corrn         Filtered State cofactor matrix Qxx
   355.4204    -0.0115       0.000033333

Epoch =   4, measurement = 355.4286
Filtered State  Corrn         Filtered State cofactor matrix Qxx
   355.4224     0.0021       0.000025000

Epoch =   5, measurement = 355.4232
Filtered State  Corrn         Filtered State cofactor matrix Qxx
   355.4226     0.0002       0.000020000
:
:
:
Epoch = 248, measurement = 355.4360
Filtered State  Corrn         Filtered State cofactor matrix Qxx
   355.4199     0.0001       0.000000403

Epoch = 249, measurement = 355.4210
Filtered State  Corrn         Filtered State cofactor matrix Qxx
   355.4199     0.0000       0.000000402

Epoch = 250, measurement = 355.4204
Filtered State  Corrn         Filtered State cofactor matrix Qxx
   355.4199     0.0000       0.000000400
```

MATLAB function *kalship3.m*

```
function kalship3(filename)
%
% kalship3   This function implements a Kalman Filter to estimate the
%   POSITION and VELOCITY of a ship in a navigation channel.
%   The observations are distances to the ship at regular time intervals
%   from three known locations.
%
% e.g., kalship3('d:\projects\kalman\exercise\kalshipdata3.txt');

%============================================================================
% Function:  kalship3
%
% Author:
%  Rod Deakin,
%  BONBEACH, VIC 3196,
%  AUSTRALIA
%  email: randm.deakin@gmail.com
%
% Date:
%  Version 1.0  17 April  2006
%  Version 1.1  29 August 2015
%
% Remarks:
%  This function implements a Kalman Filter to estimate the POSITION and
%  VELOCITY of a ship moving at a near constant velocity in a channel.
%  The observations are horizontal distances to three fixed stations
%  taken at regular time intervals.
%  The observations are contained in an ASCII data file, an example of
%  which is shown below (d:\projects\kalman\Kalshipdata3.txt)
%
%-----start of data file d:\projects\kalman\exercise\Kalshipdata3.txt------
%
% Kalman Filter Navigation problem
%
% A ship is moving at a constant velocity with distance measurements
% every 60 seconds to three shore-based beacons A, B and C.
%
% Data
%       Distances to beacons
% Epoch   A       B       C
%   1   4249.7  7768.6  7721.1
%   2   3876.1  7321.4  7288.5
%   3   3518.4  6872.2  6857.6
%   :       :       :       :
%   :       :       :       :
%   :       :       :       :
%   19  5366.4  1959.6  2819.7
%   20  5785.0  2182.8  3023.5
%----------------------end of data file--------------------------------
%
% References:
%  [1] Deakin, R.E., 2015, Least Squares and Kalman Filtering,
%        Lecture Notes, September 2015.
%  [2] Deakin, R.E., 2006, The Kalman Filter and Surveying Applications,
%        Presented at the Victorian Regional Surveying Conference, Mildura,
%        23-25 June 2006.
%
% Arrays:
%  B         - Design matrix, dimensions (m,n)
%  Distance  - vector of distances from start at each epoch
%  Ef        - vector of east coords of fixed stations (beacons)
%  epoch     - vector of epoch numbers (integers)
%  H         - coefficient matrix of System Driving Noise, dimensions (n,u)
%  Heading   - vector of headings (bearings from north) at each epoch
%  I         - Identity matrix, dimensions (n,n)
%  K         - Gain matrix, dimension (n,m)
%  meas      - array of measured distances from the ship to the beacons (metres)
%  Nf        - vector of north coords of fixed stations (beacons)
%  Q         - cofactor matrix of measurements, dimension is (m,m)
%  Qmm       - cofactor matrix of Dynamic Model, dimensions (n,n)
%  Qww       - cofactor matrix of System Driving Noise, dimensions (u,u)
%  Qxx       - cofactor matrix of State vector, dimensions (n,n)
%  T         - Transition matrix, dimensions (n,n)
%  U         - cofactor Update matrix, dimensions (n,n)
%  Velocity  - vector of velocities at each epoch
%  xhat      - State vector, dimension is (n,epochs)
%
% Variables
%  cj,dj     - distance  coefficients for observation j
%  d         - distance
%  d2        - distance squared
%  dt        - time difference in seconds between epochs
%  dE,dN     - difference in east and north coordinates
%  j,k       - integer counters
%  m         - number of measurements at each epoch
%  n         - number of parameters in the State vector 'xhat'
%  s2_aE     - variance of east  acceleration (m^2/s^4)
%  s2_aN     - variance of north acceleration (m^2/s^4)
%  s2_Dist   - variance of observed distance(m^2)
%  u         - number of elements in System Driving Noise vector
%  vE,vN     - east and north velocities
%============================================================================

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Set the coordinates of the fixed stations, the beacons A, B and C. %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Ef = [10000 13880 15550];
Nf = [10000 11250 7160];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Read in the data from a file using function textread. %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Read the epoch number into array 'epoch' and the distances to the
% three beacons into matrix 'meas'.  Distances to beacons A, B and C
% are in matrix 'meas' in columns 1, 2 and 3
[epoch,meas(:,1),meas(:,2),meas(:,3)]=textread(filename,'%d %f %f %f','headerlines',8);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Set estimates of variances of measurements and System Driving Noise %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Set the estimate of the variance of measurements.
s2_meas = 1.0;  % variance of distance (m2)
% Set the estimates of variances of System Driving Noise.
s2_aE = 0.017;
s2_aN = 0.017;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Set the Epoch update rate. %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
dt = 60;  % update rate in seconds of time

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Define the Kalman filter matrices. %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n = 4;  % number of elements in State vector
m = 3;  % number of measurements at each epoch
u = 2;  % number of elements in System Driving Noise vector

% The State vector has 4 elements; the east and north cordinates E,N, and
% the east and north velocities vE,vN.
% There are three measurements at each epoch (to beacons A, B and C).
% There are two elements in the system driving noise vector (accel. East
% and accel. North).

% Determine the number of measurement epochs.
epochs = length(epoch);
% Initialize the State vector.
xhat = zeros(n,epochs);
% Initialize the corrections to State vector.
corrn = zeros(n,epochs);
% Set the State Transition matrix T
T = eye(n);
T(1,3) = dt;
T(2,4) = dt;
% Set the cofactor matrix of the measurements Q
Q = zeros(m,m);
for m = 1:m
  Q(m,m) = s2_meas;  % diagonal elements of Q = variance of distance
end
% Set the cofactor matrix of the System Driving Noise Qww
Qww = zeros(u,u);
Qww(1,1) = s2_aE;
Qww(2,2) = s2_aN;
% Set the coefficient matrix of System Driving Noise H
H = zeros(n,u);
H(1,1) = (dt^2)/2;
H(2,2) = H(1,1);
H(3,1) = dt;
H(4,2) = H(3,1);
% Compute cofactor matrix of Dynamic Model by propagation of variances
Qmm = H*Qww*H';
% Set the starting estimate of the State vector.
% This will be the filtered Sate at epoch t2.
E1 = 7875.0;
N1 = 6319.392;
vE = 7;
vN = 3;

% Epoch t1
xhat(1,1) = E1;
xhat(2,1) = N1;
xhat(3,1) = vE;
xhat(4,1) = vN;
% Set the State cofactor matrix Qxx.
% This will be the filtered State cofactor matrix at epoch t2.
Qxx = zeros(n,n);
Qxx(1,1) = 20.0;
Qxx(2,2) = 20.0;
Qxx(3,3) = 0.5;
Qxx(4,4) = 0.5;
% Initialize the Gain matrix
K  = zeros(n,m);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Start the Kalman Filter at epoch t2. %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for k = 2:epochs
  % Compute the predicted State vector.
  xhat(:,k) = T*xhat(:,k-1);
  % Compute the predicted State cofactor matrix
  Qxx = T*Qxx*T' + Qmm;
  % Compute the Design matrix B
  B  = zeros(m,n);
  f  = zeros(m,1);
  for j = 1:m
    % coordinate differences Pi to Pk
    dE = xhat(1,k)-Ef(j);
    dN = xhat(2,k)-Nf(j);
    % Computed distance Pi to Pk
    d2 = dE^2 + dN^2;
    d  = sqrt(d2);
    % Distance coefficients
    cj =  dN/d;
    dj =  dE/d;
    % Set the elements of B
    B(j,1) = -dj;
    B(j,2) = -cj;
    % Compute the numeric terms (computed - observed)
    f(j,1) = d - meas(k,j);
  end
  % Compute the Gain matrix K
  K = Qxx*B'/(Q + B*Qxx*B');
  % Compute corrections to predicted State
  corrn(:,k) = K*f;
  % Compute the filtered State vector
  xhat(:,k) = xhat(:,k) + corrn(:,k);
  % Compute the cofactor update matrix
  I = eye(n);
  U = I - K*B;
  % Compute the filtered State cofactor matrix
  Qxx = U*Qxx;
  % Print the State vector, corrections and filtered State cofactor matrix
  fprintf('\n\nepoch = %3d',k);
  fprintf('\nFiltered State  Corrns        Filtered State cofactor matrix Qxx');
  for i=1:n
    fprintf('\n  %9.3f %10.3f      ',xhat(i,k),corrn(i,k));
    for j=1:n
      fprintf('%10.6f',Qxx(i,j));
    end
  end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%
% End of Kalman filter. %%
%%%%%%%%%%%%%%%%%%%%%%%%%%

fprintf('\n\n');

% Compute distance from start and Velocity
% and get corrections to east, north, velocity east and
% velocity north from the array of corrections.

Distance = zeros(epochs,1);
Velocity = zeros(epochs,1);
Heading  = zeros(epochs,1);
corrnE   = zeros(epochs,1);
corrnN   = zeros(epochs,1);
corrnvE  = zeros(epochs,1);
corrnvN  = zeros(epochs,1);

d2r = 180/pi; % degree to radian conversion factor d2r = 57.29577951..
for k=1:epochs
    if k >1
        % coordinate differences P(k-1) to P(k)
        dE = xhat(1,k)-xhat(1,k-1);
        dN = xhat(2,k)-xhat(2,k-1);
        % Computed distance P(k-1) to P(k)
        d = sqrt(dE^2 + dN^2);
        Distance(k,1) = Distance(k-1,1) + d;
    end
    angle = atan2(xhat(3,k),xhat(4,k))*d2r;
    if angle < 0
        angle = angle + 360;
    end
    Velocity(k,1) = sqrt(xhat(3,k)^2 + xhat(4,k)^2);
    Heading(k,1)  = angle;
    corrnE(k,1)   = corrn(1,k);
    corrnN(k,1)   = corrn(2,k);
    corrnvE(k,1)  = corrn(3,k);
    corrnvN(k,1)  = corrn(4,k);
end

% Print the filtered values

fprintf('Filtered Values');
fprintf('\nEpoch  Distance  Velocity  Heading');
for k=1:epochs
  fprintf('\n%3d %11.3f %8.3f %8.3f',k,Distance(k,1),Velocity(k,1),Heading(k,1));
end

fprintf('\n\n');

%-------------------------
% create plot of velocities
%-------------------------
figure(1);
clf;
grid on;
plot(epoch,Velocity,'bo-');
title('Kalman Filter Velocities');
xlabel('Epoch');
ylabel('Velocity (m/s)');

%-----------------------------------------
% create plot of East coordinate corrections
%-----------------------------------------
figure(2);
clf;
grid on;
plot(epoch,corrnE,'bo-');
title('Kalman Filter corrections to East coords');
xlabel('Epoch');
ylabel('Corrections (m)');

%-----------------------------------------
% create plot of North coordinate corrections
%-----------------------------------------
figure(3);
clf;
grid on;
plot(epoch,corrnN,'bo-');
title('Kalman Filter corrections to North coords');
xlabel('Epoch');
ylabel('Corrections (m)');
```

Help message for MATLAB function *kalship3.m*

```
>> help kalship3
  kalship3   This function implements a Kalman Filter to estimate the
    POSITION and VELOCITY of a ship in a navigation channel.
    The observations are distances to the ship at regular time intervals
    from three known locations.

  e.g., kalship3('d:\projects\kalman\exercise\kalshipdata3.txt');

>>
```

Data file (`d:\projects\kalman\exercise\Kalshipdata3.txt`) for MATLAB function *kalship3.m*

```
% Kalman Filter Navigation problem
%
% A ship is moving at a constant velocity with distance measurements
% every 60 seconds to three shore-based beacons A, B and C.
%
% Data
%      Distances to beacons
% Epoch   A       B       C
    1   4249.7  7768.6  7721.1
    2   3876.1  7321.4  7288.5
    3   3518.4  6872.2  6857.6
    4   3193.3  6426.0  6429.1
    5   2903.6  5982.6  6009.7
    6   2664.0  5543.2  5596.6
    7   2490.9  5107.7  5191.5
    8   2392.9  4678.9  4797.1
    9   2383.2  4253.4  4417.8
    10  2463.0  3841.7  4050.9
    11  2623.2  3435.6  3709.9
    12  2849.0  3054.2  3395.8
    13  3126.7  2692.9  3119.4
    14  3446.9  2366.6  2891.1
    15  3793.4  2096.4  2724.4
    16  4166.0  1900.6  2630.9
    17  4552.2  1804.7  2610.2
    18  4956.2  1824.8  2677.4
    19  5366.4  1959.6  2819.7
    20  5785.0  2182.8  3023.5
```

Output from MATLAB function *kalship3.m*

```
>> kalship3('d:\projects\kalman\exercise\kalshipdata3.txt');


epoch =   2
Filtered State  Corrns        Filtered State cofactor matrix Qxx
   8289.594     -5.406        1.009225 -0.797965  0.033097 -0.026169
   6521.882     22.490       -0.797965  1.439797 -0.026169  0.047217
      6.823     -0.177        0.033097 -0.026169  0.506780 -0.000858
      3.738      0.738       -0.026169  0.047217 -0.000858  0.507243

epoch =   3
Filtered State  Corrns        Filtered State cofactor matrix Qxx
   8705.780      6.823        0.926924 -0.643621  0.030398 -0.021105
   6727.944    -18.189       -0.643621  1.218371 -0.021105  0.039955
      7.046      0.224        0.030398 -0.021105  0.494715 -0.004015
      3.141     -0.596       -0.021105  0.039955 -0.004015  0.496822

epoch =   4
Filtered State  Corrns        Filtered State cofactor matrix Qxx
   9124.759     -3.808        0.863463 -0.498398  0.028327 -0.016346
   6928.604     12.198       -0.498398  1.011477 -0.016346  0.033180
      6.922     -0.125        0.028327 -0.016346  0.483077 -0.006336
      3.541      0.400       -0.016346  0.033180 -0.006336  0.486133
```

```
epoch =   5
Filtered State  Corrns        Filtered State cofactor matrix Qxx
   9540.095      0.042        0.798512 -0.363666  0.026205 -0.011929
   7132.756     -8.319       -0.363666  0.856483 -0.011929  0.028105
      6.923      0.001        0.026205 -0.011929  0.471881 -0.007919
      3.268     -0.273       -0.011929  0.028105 -0.007919  0.475282
:
:
:
epoch =  17
Filtered State  Corrns        Filtered State cofactor matrix Qxx
  14531.436     -5.091        1.097853  0.503116  0.036151  0.016564
   9565.143      0.684        0.503116  0.809750  0.016567  0.026660
      6.827     -0.168        0.036151  0.016567  0.373179  0.001640
      3.346      0.023        0.016564  0.026660  0.001640  0.377365

epoch =  18
Filtered State  Corrns        Filtered State cofactor matrix Qxx
  14950.377      9.319        0.955699  0.439824  0.031474  0.014481
   9770.483      4.566        0.439824  0.822675  0.014483  0.027090
      7.134      0.307        0.031474  0.014483  0.368845  0.003650
      3.497      0.150        0.014481  0.027090  0.003650  0.371765

epoch =  19
Filtered State  Corrns        Filtered State cofactor matrix Qxx
  15366.544    -11.869        0.732074  0.288579  0.024112  0.009501
   9973.570     -6.708        0.288579  0.820826  0.009502  0.027033
      6.743     -0.391        0.024112  0.009502  0.364017  0.005200
      3.276     -0.221        0.009501  0.027033  0.005200  0.366470

epoch =  20
Filtered State  Corrns        Filtered State cofactor matrix Qxx
  15781.273     10.148        0.598119  0.158080  0.019704  0.005204
  10175.278      5.167        0.158080  0.847313  0.005203  0.027911
      7.077      0.334        0.019704  0.005203  0.358551  0.006055
      3.446      0.170        0.005204  0.027911  0.006055  0.361445

Filtered Values
Epoch  Distance  Velocity  Heading
  1       0.000    7.616   66.801
  2     461.400    7.779   61.286
  3     925.806    7.715   65.975
  4    1390.357    7.775   62.905
  5    1853.155    7.656   64.729
  6    2315.773    7.765   63.208
  7    2778.387    7.660   65.206
  8    3240.322    7.741   62.817
  9    3703.373    7.698   64.889
 10    4165.683    7.713   63.805
 11    4631.259    7.805   63.717
 12    5091.795    7.550   64.711
 13    5555.396    7.900   63.076
 14    6020.782    7.618   64.761
 15    6481.164    7.728   63.223
 16    6945.300    7.744   64.584
 17    7405.657    7.603   63.888
 18    7872.215    7.945   63.889
 19    8335.291    7.497   64.090
 20    8796.471    7.872   64.039

>>
```

MATLAB function *global_warming_filter.m*

```
function glogbal_warming_filter
%
% function to use a Kalman Filter to estimate the signal in a sequence of
% temperature anomalies.  Two data files are available:
%   (1) Global Land-Ocean Temperature Anomalies (deg C) reative to
%       1951-1980 average
%       d:\temp\Anomalies_NASA_1880_2014.txt
%       and
%   (2) Contiguous 48 U.S. Surface Air Temperature Anomalies (deg C)
%       relative to 1951-1980 average.
%       d:\temp\Anomalies_US_Surface_Air_Temp_1880_2014.txt
%
% Both data sets are derived from datasets available from the National
% Aeronautics and Space Administration (NASA) website at:
% http://data.giss.nasa.gov/gistemp/ under Datasets & Images

%==========================================================================
% Function:  global_warming_filter
%
% Author:
%  Rod Deakin,
%  BONBEACH, VIC, 3196
%  AUSTRALIA
%  email: randm.deakin@gmail.com
%
% Date:
%  Version 1.0  06 April 2015
%  Version 1.1  01 September 2015
%    Moving Average filter added
%
% Remarks:
%  This function uses a Kalman Filter to estimate the signal in global
%  warming temperature anomalies.
%
% References:
%  [1] Deakin, R.E., 2015, Least Squares and Kalman Filtering,
%        Lecture Notes, September 2015.
%  [2] Deakin, R.E., 2006, The Kalman Filter and Surveying Applications,
%        Lecture Notes, School of Mathematical and Geospatial Sciences,
%        RMIT University, June 2006, 30 pages.
%  [3] Deakin, R.E., 2006, The Kalman Filter: A Look Behind the Scene,
%        Presented at the Regional Surveying Conference, Mildura, 23-25
%        June 2006, 12 pages.
%
% Arrays:
%  Anomaly   - vector of temperature anomalies
%  B         - Design matrix, dimensions (m,n)
%  b5        - 5-element vector for Moving Average filter
%  corrn     - array of corrections to State vector, dimensions (n,epochs)
%  H         - coefficient matrix of System Driving Noise, dimensions (n,u)
%  I         - Identity matrix, dimensions (n,n)
%  K         - Gain matrix, dimensions (n,m)
%  meas      - vector of measurements, dimensions (epochs,1)
%  Q         - cofactor matrix of measurements, dimensions (n,n)
%  Qmm       - cofactor matrix of Secondary Model, dimensions (n,n)
%  Qww       - cofactor matrix of System Driving Noise, dimensions (n,n)
%  Qxx       - cofactor matrix of State vector, dimensions (n,n)
%  std_xhat  - vector of standard deviations of the filtered State
%  T         - Transition matrix, dimensions (n,n)
%  Temp      - vector of global temperatures
%  U         - cofactor update matrix, dimensions (n,n)
%  xhat      - array containing the State vector at each epoch,
%              dimensions (n,epochs)
%  Year      - vector of years
%  y5        - vector of values for a 5-year moving average filter,
%              dimensions(epochs,1)
%
% Variables:
%  epochs    - number of years
%  i,j,k     - integer counters
%  m         - number of measurements at particular epoch
%  n         - number of parameters in the State vector 'xhat'
%  std_meas  - standard deviation of measurements
%  u         - number of parameters in the System Driving Noise vector
%==========================================================================

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Read data from text file %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%-------------------------------------------------------------------------
% 1. Call the User Interface (UI) to choose the input data file name
% 2. Concatenate strings to give the path and file name of the input file
% 3. Strip off the extension from the file name to give the rootName
% 4. Add extension ".out" to rootName to give the output filename
% 5. Concatenate strings to give the path and file name of the output file
%-------------------------------------------------------------------------
filepath = strcat('d:\temp\','*.txt');
[infilename,inpathname] = uigetfile(filepath);
infilepath = strcat(inpathname,infilename);
rootName   = strtok(infilename,'.');
outfilename = strcat(rootName,'.out');
outfilepath = strcat(inpathname,outfilename);

%----------------------------------------------------------
% 1. Load the data into an array whose name is the rootName
% 2. set fileTemp = rootName
% 3. Copy columns of data into individual arrays
%----------------------------------------------------------
load(infilepath);
fileTemp = eval(rootName);
Year    = fileTemp(:,1);
Anomaly = fileTemp(:,2);

% determine the number of measurement epochs
epochs = length(Year);

% Set vector of Global Surface Temperatures where the Global Surface
% Temperature = Anomaly + 14.0 and 14.0 is the mean of the Temperatures for
% the period 1951-1980.
Temp = Anomaly + 14.0;  % NASA dataset

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Define the Kalman filter matrices %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
n = 1;  % number of elements in State vector
m = 1;  % number of measurements at each epoch
u = 1;  % number of elements in System Driving Noise vector
% Initialize the State vector.
xhat = zeros(n,epochs);
% Initialize the corrections to State vector.
corrn = zeros(n,epochs);
% Set the State Transition matrix T
T = eye(n);
T(1,1) = 1.0;
% Set the cofactor matrix of the System Driving Noise Qww
Qww = zeros(u,u);
Qww(1,1) = 0.01;
% Set the coefficient matrix of System Driving Noise H
H = zeros(n,u);
H(1,1) = 1.0;
```

```
% Compute cofactor matrix of Dynamic Model by propagation of variances
Qmm = H*Qww*H';
% Initialize vector of measurements
meas = zeros(epochs,1);
% Set the measurement vector to either Temperatures or Temp Anomalies
%meas = Temp;
meas = Anomaly;
% Set the standard deviation of measurements
std_meas = sqrt(0.5);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Set Cofactor matrices and State vector to initial values %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Set the cofactor matrix of the measurements Q
Q = zeros(m,m);
Q(1,1)  = std_meas^2;
% Set the State cofactor matrix Qxx.  This will be the filtered
% State cofactor matrix at epoch t2.
Qxx = zeros(n,n);
Qxx(1,1) = std_meas^2;
% Set the starting estimate of the State vector for Epoch 1
% This will be the filtered Sate at epoch t2
xhat(1,1) = meas(1,1);
% Initialize the Gain matrix
K = zeros(n,m);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Start the Kalman Filter at epoch t2 %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for k = 2:epochs
  % Compute the predicted State vector.
  xhat(:,k) = T*xhat(:,k-1);
  % Compute the predicted State cofactor matrix
  Qxx = T*Qxx*T' + Qmm;
  % Set the elements of the design matrix B
  B = zeros(m,n);
  B(1,1) = -1;
  % Compute the Gain matrix K
  K = Qxx*B'/(Q + B*Qxx*B');
  % Compute corrections to predicted State
  corrn(:,k) = K*(-meas(k) - B*xhat(:,k));
  % Compute the filtered State vector
  xhat(:,k) = xhat(:,k) + corrn(:,k);
  % Compute the cofactor update matrix
  I = eye(n);
  U = I - K*B;
  % Compute the filtered State cofactor matrix
  Qxx = U*Qxx;
  std_xhat(k) = sqrt(Qxx(1,1));

  % Print the State vector, corrections and filtered State cofactor matrix
  fprintf('\n\nEpoch = %3d, Year = %4d, measurement = %8.4f',k,Year(k),meas(k));
  fprintf('\nFiltered State  Corrn         Filtered State cofactor matrix Qxx');
  for i=1:u
      fprintf('\n  %9.4f %10.4f      ',xhat(i,k),corrn(i,k));
      for j=1:u
          fprintf('%12.9f',Qxx(i,j));
      end
  end
end
%%%%%%%%%%%%%%%%%%%%%%%%%
% End of Kalman filter %%
%%%%%%%%%%%%%%%%%%%%%%%%%
fprintf('\n\n');
```

```
%%%%%%%%%%%%%%
% plot data %%
%%%%%%%%%%%%%%

% Figure 1: Plot of Temperature Anomalies
figure(1);
clf(1);
hold on;
grid on;
box on;
% set scaling for the x- and y-axes
axis([1870 2020 -1.5 1.5])
%axis square;
% plot the data
plot(Year,Anomaly,'bo','MarkerSize',3);
% anotate the plot
%title('Global Land-Ocean Temperature Anomalies 1880-2014')
title('Contiguous 48 U.S. Surface Air Temperature Anomalies 1880-2014')
xlabel('Year');
ylabel('Temp Anomaly (deg C) relative to 1951-1980 average');

% Figure 2: create a figure window with two plots.  The upper plot shows
% the measurements and the filtered state.  The lower plot shows the
% standard deviation of the filtered state
figure(2);
clf(2);
subplot(2,1,1);
hold on;
grid on;
box on;
plot(meas,'k.','MarkerSize',5);
plot(xhat,'k');
title('Kalman Filter estimate');
xlabel('epoch');
ylabel('state [deg C]');

subplot(2,1,2);
hold on;
grid on;
box on;
plot(std_xhat,'k');
title('Standard deviation of estimate');
xlabel('epoch');
ylabel('std [deg C]');

% Figure 3: Plot of Kalman Filter estimate
figure(3);
clf(3);
hold on;
grid on;
box on;

% set scaling for the x- and y-axes
axis([1870 2020 -1.5 1.5])
% plot the data
plot(Year,Anomaly,'bo','MarkerSize',3);
plot(Year,xhat,'k');
% anotate the plot
%title('Global Land-Ocean Temperature Anomalies 1880-2014')
title({'Contiguous 48 U.S. Surface Air Temperature Anomalies 1880-2014';'with signal (Kalman
Filter)'});
xlabel('Year');
ylabel('Temp Anomaly (deg C) relative to 1951-1980 average');
```

```
% Figure 4: Plot of standard deviation of Kalman Filter estimate
figure(4);
clf(4);
hold on;
grid on;
plot(std_xhat,'k');
hold off;
title('Standard deviation of estimate');
xlabel('epoch');
ylabel('std [deg C]');

%%%%%%%%%%%%%%%%%%%%%%%%%%
% Moving Average Filter %%
%%%%%%%%%%%%%%%%%%%%%%%%%%
% 5-year moving average
b5  = [1/5 1/5 1/5 1/5 1/5];
y5  = filter(b5,1,Anomaly);

% centre the averages about mid-values
for k = 3:epochs-2
    y5(k) = y5(k+2);
end
y5(1) = 0;
y5(2) = 0;
y5(epochs-1) = 0;
y5(epochs)   = 0;

% Figure 5: Moving average filter
figure(5);
clf(5);
hold on;
grid on;
box on;
% set scaling for the x- and y-axes
axis([1870 2020 -1.5 1.5])
% plot the data
plot(Year,Anomaly,'bo','MarkerSize',3);
% plot centred 5-year moving average
plot(Year(3:epochs-2),y5(3:epochs-2),'k-','LineWidth',1);
% anotate the plot
title({'Contiguous 48 U.S. Surface Air Temperature Anomalies 1880-2014';'with centred 5-Year
Moving Average'});
xlabel('Year');
ylabel('Temp Anomaly (deg C) relative to 1951-1980 average');
```

Help message for MATLAB function *linear_regression_CLS.m*

```
>> help global_warming_filter
  function to use a Kalman Filter to estimate the signal in a sequence of
  temperature anomalies.  Two data files are available:
    (1) Global Land-Ocean Temperature Anomalies (deg C) reative to
        1951-1980 average
        d:\temp\Anomalies_NASA_1880_2014.txt
        and
    (2) Contiguous 48 U.S. Surface Air Temperature Anomalies (deg C)
        relative to 1951-1980 average.
        d:\temp\Anomalies_US_Surface_Air_Temp_1880_2014.txt

  Both data sets are derived from datasets available from the National
  Aeronautics and Space Administration (NASA) website at:
  http://data.giss.nasa.gov/gistemp/ under Datasets & Images

>>
```

Data file (`d:\temp\Anomalies_US_Surface_Air_Temp_1880_2014.txt`) for MATLAB function *global_warming_filter.m*

```
%Contiguous 48 U.S. Surface Air Temperature Anomaly (C)
%-------------------------------------------------------
%Year   Annual_Mean
1880    -0.4656
1881     0.0693
1882    -0.0067
1883    -0.8181
1884    -0.6041
1885    -0.6516
1886    -0.4563
1887    -0.2135
:
:
:
2010     0.5946
2011     0.6747
2012     1.8681
2013     0.2093
2014     0.2816
% compiled from http://data.giss.nasa.gov/gistemp/graphs_v3/Fig.D.txt
```

Output from MATLAB function *global_warming_filter.m*

```
>> global_warming_filter

Epoch =   2, Year = 1881, measurement =   0.0693
Filtered State  Corrn         Filtered State cofactor matrix Qxx
    -0.1955     0.2701       0.252475248

Epoch =   3, Year = 1882, measurement =  -0.0067
Filtered State  Corrn         Filtered State cofactor matrix Qxx
    -0.1305     0.0650       0.172120504

Epoch =   4, Year = 1883, measurement =  -0.8181
Filtered State  Corrn         Filtered State cofactor matrix Qxx
    -0.3141    -0.1836       0.133495843
:
:
:
Epoch = 133, Year = 2012, measurement =   1.8681
Filtered State  Corrn         Filtered State cofactor matrix Qxx
     0.7864     0.1642       0.065887234

Epoch = 134, Year = 2013, measurement =   0.2093
Filtered State  Corrn         Filtered State cofactor matrix Qxx
     0.7104    -0.0760       0.065887234

Epoch = 135, Year = 2014, measurement =   0.2816
Filtered State  Corrn         Filtered State cofactor matrix Qxx
     0.6539    -0.0565       0.065887234
```