Parameter Estimation

T. Hesketh

March, 2011

Introduction

Parameter estimation is one part of system identification, which includes determination of model structure, experiment design, parameter estimation and validation. For present purposes, it is assumed that structure is determined by the operator and that the model is linear in the parameters and restricted to discrete transfer functions.

Experiment design requires selection of the actuation. This is simpler in open loop, but poses difficulties in closed loop estimation. Recent work by Graham Goodwin has made the requirements for closed loop parameter estimation a lot clearer.

Least squares is a basic technique for parameter estimation. We will concentrate on least squares because the concepts are readily grasped, and while more sophisticated methods exist, it is probably more important to understand some basics thoroughly, and to consider some of the closed loop identification issues.

Much of the following treatment is taken from Astrom and Wittenmark, 1989.

Least Squares Models and Regression

Least squares is particularly appropriate for a mathematical model that can be written as:

$$y(t) = \phi_1(t)\theta_1 + \phi_2(t)\theta_2 + \ldots + \phi_n(t)\theta_n = \phi(t)\theta$$

where y is the measurement, $\theta_1, \theta_2, \ldots, \theta_n$ are unknown parameters (to be estimated) and $\phi_1, \phi_2, \ldots, \phi_n$ are known functions.

Example

Consider a process with the transfer function

$$y(t) = \frac{b_1 q^{-1} + b_2 q^{-2}}{1 + a_1 q^{-1} + a_2 q^{-2}} u(t)$$

which may be re-expressed as

$$y(t) = b_1 u(t-1) + b_2 u(t-2) - a_1 y(t-1) - a_2 y(t-2)$$

and finally

$$y(t) = \begin{bmatrix} u(t-1) & u(t-2) & y(t-1) & y(t-2) \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ -a_1 \\ -a_2 \end{bmatrix}$$

We can write many simultaneous equations from this, predicting $y(t),\ y(t-1),\ y(t-2) \ldots$.

Least squares involves solving a set of overdetermined linear simultaneous equations of the form

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & \ldots & x_{1n} \\ x_{21} & x_{22} & \ldots & x_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ x_{m1} & x_{m2} & \ldots & x_{mn} \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix}$$

$$Y = X\theta$$

Least squares aims to find the solution $\hat{\theta}$ that minimise the sum of squares of the difference between the observed data and the estimated values:

$$\hat{\theta} = \arg\min_\theta \|Y - X\theta\|_2$$

The solution is given by the normal equation

$$X^\top Y = \left[X^\top X\right] \theta$$

and

$$\hat{\theta} = \left[X^\top X\right]^{-1} X^\top Y$$

and the loss function (sum of squared residuals) is given by

$$J = \left[Y - X\hat{\theta}\right]^\top \left[Y - X\hat{\theta}\right]$$

Proof

$$\begin{aligned} J &= (Y - X\theta)^\top (Y - X\theta) \\ &= Y^\top Y - Y^\top X\theta - \theta^\top X^\top Y + \theta^\top X^\top X\theta \end{aligned}$$

Since the matrix $X^\top X$ is always non-negative definite, the function J has a minimum. The loss function is quadratic in θ. By completing the square, it is possible to find the minimum.

$$\begin{aligned} J &= Y^\top Y - Y^\top X\theta - \theta^\top X^\top Y + \theta^\top X^\top X\theta \\ &\quad + Y^\top X(X^\top X)^{-1}X^\top Y - Y^\top X(X^\top X)^{-1}X^\top Y \\ &= Y^\top(I - X(X^\top X)^{-1}X^\top)Y \\ &\quad + (\theta - (X^\top X)^{-1}X^\top Y)^\top X^\top X(\theta - (X^\top X)^{-1}X^\top Y) \end{aligned}$$

The first term on the RHS is independent of θ. The second term is always positive. The minimum is obtained for

$$\theta = \hat{\theta} = (X^\top X)^{-1}X^\top Y$$

In practice, this method is never used due to poor numerical performance. A number of methods for factorising $P = X^\top X$ exist [1] and foremost amongst these is the QR decomposition [2]

$$X = Q\begin{bmatrix} R \\ 0 \end{bmatrix} \text{ and } Q^\top Y = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}$$

and the least squares estimate is given by solving

$$R\theta = z_1$$

The loss function is

$$J = \|Y - X\hat{\theta}\|_2 = \|z_2\|_2$$

Recursive Least Squares

To derive a recursive algorithm we need the following lemma.

Matrix Inversion Lemma

Let A, C, and $C^{-1} + DA^{-1}B$ be nonsingular square matrices. Then:

$$(A + BCD)^{-1} = A^{-1} - A^{-1}B(C^{-1} + DA^{-1}B)^{-1}DA^{-1}$$

Proof By direct substitution:

[1] Steve S. Niu, D. Grant Fisher, Lennart Ljung, Sirish L. Shah, "A Tutorial on Multiple Model Least-Squares and Augmented UD Identification", December 21, 1994.

[2] Golub, G.H. & van Loan, C.F.V. (1989), "Matrix Computations", 2nd Edition, The Johns Hopkins University Press, Oxford.

$$
\begin{aligned}
(A+BCD) \qquad & (A^{-1} - A^{-1}B(C^{-1}+DA^{-1}B)^{-1}DA^{-1}) \\
= \; & I + BCDA^{-1} - B(C^{-1}+DA^{-1}B)^{-1}DA^{-1} \\
& -BCDA^{-1}B(C^{-1}+DA^{-1}B)^{-1}DA^{-1} \\
= \; & I + BCDA^{-1} - BC(C^{-1}+DA^{-1}B(C^{-1}+DA^{-1}B)(C^{-1}+DA^{-1}B)^{-1}DA^{-1} \\
= \; & I + BCDA^{-1} - BCDA^{-1} \\
= \; & I
\end{aligned}
$$

The solution to the least squares problem can now be written in a recursive form. It follows from the definition of $P = X^\top X$ that:

$$P(t)^{-1} = P(t-1)^{-1} + \phi(t)\phi\top(t)$$

the least squares estimate is given by:

$$\hat{\theta} = P(t)\left(\sum_{i=1}^{t}\phi(i)y(i)\right) = P(t)\left(\sum_{i=1}^{t-1}\phi(i)y(i) + \phi(t)y(t)\right)$$

Using the expression for $P(t)^{-1}$ gives:

$$\sum_{i=1}^{t-1}\phi(i)y(i) = P(t-1)^{-1}\hat{\theta}(t-1) = P(t)^{-1}\hat{\theta}(t-1) - \phi(t)\phi^\top(t)\theta(t\hat{-}1)$$

The estimate at time t can now be written as:

$$
\begin{aligned}
\hat{\theta}(t) &= \hat{\theta}(t-1) - P(t)\phi(t)\phi^\top(t)\hat{\theta}(t-1) + P(t)\phi(t)y(t) \\
&= \hat{\theta}(t-1) + P(t)\phi(t)\left(y(t) - \phi^\top(t)\hat{\theta}(t-1)\right) \\
&= \hat{\theta}(t-1) + K(t)\epsilon(t)
\end{aligned}
$$

where

$$
\begin{aligned}
K(t) &= P(t)\phi(t) \\
\epsilon(t) &= y(t) - \phi^\top\hat{\theta}(t-1)
\end{aligned}
$$

The residual $\epsilon(t)$ can be interpreted as the prediction error (one step ahead) of $y(t)$ based on the estimate $\hat{\theta}(t-1)$.

Applying the matrix inversion lemma to $P(t)$ and using the equation for $P(t)^{-1}$ gives:

$$
\begin{aligned}
P(t) &= \left(\Phi^\top(t)\Phi(t)\right)^{-1} = \left(\Phi^\top(t-1)\Phi(t-1) + \phi(t)\phi^\top(t)\right)^{-1} \\
&= \left(P(t-1)^{-1} + \phi(t)\phi^\top(t)\right)^{-1} \\
&= P(t-1) - P(t-1)\phi(t)\left(I + \phi^\top(t)P(t-1)\phi(t)\right)^{-1}\phi^\top(t)P(t-1)
\end{aligned}
$$

This implies that

$$K(t) = P(t)\phi(t) = P(t-1)\phi(t)\left(I + \phi^\top(t)P(t-1)\phi(t)\right)^{-1}\phi^\top P(t-1)$$

Theorem - Recursive Least Squares (RLS)

Assume that the matrix $\Phi(t)$ has full rank for all $t > t_0$. The least squares estimate $\hat{\theta}$ then satisfies the recursive equations:

$$\begin{aligned}
\hat{\theta}(t) &= \hat{\theta}(t-1) + K(t)\left(y(t) - \phi^\top(t)\hat{\theta}(t-1)\right) \\
K(t) &= P(t)\phi(t) = P(t-1)\phi(t)\left(I + \phi^\top(t)P(t-1)\phi(t)\right)^{-1} \\
P(t) &= P(t-1) - P(t-1)\phi(t)\left(I + \phi^\top(t)P(t-1)\phi(t)\right)^{-1}\phi^\top(t)P(t-1) \\
&= (I - K(t)\phi^\top(t))P(t-1)
\end{aligned}$$

---

1. The estimate $\hat{\theta}(t)$ is obtained by adding a correction to the previous estimate $\hat{\theta}(t-1)$. The correction is proportional to $y(t) - \phi^\top(t)\hat{\theta}(t-1)$.
2. The least squares estimate can be interpreted as a Kalman filter for the process

$$\begin{aligned}
\theta(t+1) &= \theta(t) \\
y(t) &= \phi^\top(t)\theta(t) + e(t)
\end{aligned}$$

Time Varying Parameters

When parameters change over time, we want to emphasise the more recent values and "forget" the older values. This can be achieved simply by replacing the least squares cost by

$$V(\theta, t) = \frac{1}{2}\sum_{i=1}^{t}\lambda^{t-i}\left(y(i) - \phi^\top(i)\theta\right)^2$$

where λ is a parameter such that $0 < \lambda \leq 1$, called the forgetting factor.

Theorem - RLS with exponential forgetting

Assume that the matrix $\Phi(t)$ has full rank for $t \geq t_0$. The parameter θ, which minimises the least squares cost, is given by:

$$\begin{aligned}
\hat{\theta}(t) &= \hat{\theta}(t-1) + K(t)\left(y(t) - \phi^\top(t)\hat{\theta}(t-1)\right) \\
K(t) &= P(t)\phi(t) = P(t-1)\phi(t)\left(\lambda I + \phi^\top(t)P(t-1)\phi(t)\right)^{-1} \\
P(t) &= P(t-1) - P(t-1)\phi(t)\left(\lambda I + \phi^\top(t)P(t-1)\phi(t)\right)^{-1}\phi^\top(t)P(t-1)/\lambda \\
&= (I - K(t)\phi^\top(t))P(t-1)/\lambda
\end{aligned}$$

Transfer Function Models

Equation Error

The process may be described by

$$(1 + A)y = Bu + \epsilon$$

The estimation error is added to the equation for the process. The least squares modelling is developed as follows:

$$\begin{aligned} y &= Bu - Ay + \epsilon \\ y &= \phi^\top \theta + \epsilon \\ \phi^\top &= \left[\begin{array}{cccccc} u(t-1) & \dots & u(t-n) & y(t-1) & \dots & y(t-n) \end{array} \right] \\ \theta^\top &= \left[\begin{array}{cccccc} b_1 & \dots & b_n & -a_1 & \dots & -a_n \end{array} \right] \end{aligned}$$

Output Error

In many situations, the output error form of the equations is preferable. In this case, the estimation error is added to the process output:

$$\begin{aligned} (1 + A)\hat{y} &= Bu \\ y &= \hat{y} + \epsilon \\ y &= Bu - A\hat{y} + \epsilon \\ y &= \phi^\top \theta + \epsilon \\ \phi^\top &= \left[\begin{array}{cccccc} u(t-1) & \dots & u(t-n) & \hat{y}(t-1) & \dots & \hat{y}(t-n) \end{array} \right] \\ \theta^\top &= \left[\begin{array}{cccccc} b_1 & \dots & b_n & -a_1 & \dots & -a_n \end{array} \right] \end{aligned}$$

Recursive Estimation

The recursive algorithms are quite similar. They can be described by the equations:

$$\begin{aligned} \hat{\theta}(t) &= \hat{\theta}(t-1) + P(t)\phi(t-1)\epsilon(t) \\ \epsilon(t) &= y(t) - \phi^\top(t-1)\hat{\theta}(t-1) \\ P(t) &= \frac{1}{\lambda}\left(P(t-1) - \frac{P(t-1)\phi(t-1)\phi^\top(t-1)P(t-1)}{\lambda + \phi^\top(t-1)P(t-1)\phi(t-1)} \right) \end{aligned}$$

ϕ is different for equation error and output error methods.

Square Root Methods

Numerical accuracy is sacrificed when solving a least squares problem using the normal equations, because measured values are squared unnecessarily. The basic idea is as follows:

$$E = Y - \Phi\theta$$

An orthogonal transformation T does not change the norm of the error.

$$\tilde{E} = TE = TY - T\Phi\theta$$

Choose the transformation T so that $T\Phi$ is upper triangular. Then:

$$\begin{bmatrix} \tilde{e_1} \\ \tilde{e}_2 \end{bmatrix} = \begin{bmatrix} \tilde{y}_1 \\ \tilde{y}_2 \end{bmatrix} - \begin{bmatrix} \tilde{\Phi}_1 \\ 0 \end{bmatrix} \theta$$

where $\tilde{\Phi}_1$ i supper triangular. The least squares estimate is then:

$$\hat{\theta} = \tilde{\Phi}_1^{-1} \tilde{y}_1$$

This method is called a *square root method* because it works with Φ, or the square root of $\Phi^\top \Phi$.

We need the following theorem, offered without proof, to develop a recursive square root method for least squares estimation.

Theorem - Conditional mean values and covariances

The vectors x and y are jointly Gaussian random variables with mean values

$$E \begin{bmatrix} y \\ x \end{bmatrix} = \begin{bmatrix} m_y \\ m_x \end{bmatrix}$$

and covariance

$$\text{cov} \begin{bmatrix} y \\ x \end{bmatrix} = \begin{bmatrix} R_y & R_{yx} \\ R_{xy} & R_x \end{bmatrix} = R$$

where $R_{xy} = R_{yx}^\top$. The conditional mean value of x given y is Gaussian with mean

$$E(x|y) = m_x + R_{xy} R_y^{-1} (y - m_y)$$

and covariance

$$\text{cov}(x|y) = R_{x|y} = R_x - R_{xy} R_y^{-1} R_{yx}$$

R can be factorised as

$$R = \rho \begin{bmatrix} 1 & 0 \\ K & L_x \end{bmatrix} \begin{bmatrix} D_y & 0 \\ 0 & D_x \end{bmatrix} \begin{bmatrix} 1 & 0 \\ K & L_x \end{bmatrix}^\top$$

where D_x and D_y are diagonal matrices and L_x is lower triangular. Then

$$\begin{aligned} R_{xy} R_y^{-1} &= K \quad \text{and} \\ R_{x|y} &= \rho L_x D_x L_x^\top \end{aligned}$$

Recursive Estimation

θ has an initial value θ_0. A linear observation

$$y = \phi^\top \theta + e$$

is made. The new estimate for θ is given by the conditional mean $E(\theta|y)$.

$$R = \begin{bmatrix} \phi^\top P \phi & \phi^\top P \\ P\phi P & \end{bmatrix} + \begin{bmatrix} \sigma^2 & 0 \\ 0 & 0 \end{bmatrix}$$

With the factorisation $P = LDL^\top$

$$\begin{aligned} R &= \begin{bmatrix} \phi^\top LDL^\top \phi + \sigma^2 & \phi^\top LDL^\top \\ LDL^\top \phi & LDL^\top \end{bmatrix} \\ &= \begin{bmatrix} 1 & \phi^\top L \\ 0 & L \end{bmatrix} \begin{bmatrix} \sigma^2 & 0 \\ 0 & D \end{bmatrix} \begin{bmatrix} 1 & 0 \\ L^\top \phi & L^\top \end{bmatrix} \end{aligned}$$

If this matrix can be transformed to

$$R = \begin{bmatrix} 1 & 0 \\ K & \tilde{L} \end{bmatrix} \begin{bmatrix} \tilde{\sigma}^2 & 0 \\ 0 & \tilde{D} \end{bmatrix} \begin{bmatrix} 1 & K^\top \\ 0 & \tilde{L}^\top \end{bmatrix}$$

the above theorem can be used to obtain the recusive estimate as

$$\hat{\theta} = \theta_0 + K(y - \phi^\top \theta)$$

with covariance

$$P = \tilde{L}\tilde{D}\tilde{L}^\top$$

The method for achieving this is known as *dyadic decomposition* and was described by Bierman, 1977.

The following code is a C version of Fortran code reported by Astrom and Wittenmark (1989) and taken from Bierman (1977), which performs least squares estimation using the $UDU^\top$ factorisation. Matrices are stored as single dimensioned vectors.

```
void ud(float y, float d[], float u[], float k[], float theta[],
         int npar, float e, float lambda)
{
// y     measured value
// phi   regression, most recent in phi[0]
// u,d   factors of r
// k     kalman gain vector
// theta parameter estimates vector
// npar  number of parameters, length of phi,d,k,theta
// e     prediction error e = y - phi' * theta
//lambda forgetting factor
    int i, j, kf, ku;
    float fj, vj, alphaj, ajlast, pj, w;

    e = y;
```

```
    for (i = 0; i < npar; i++) e -= theta[i] * phi[i];
    fj = phi[0];
    vj = d[0] * fj;
    k[0] = vj;
    alphaj = 1.0 + vj * fj;
    d[0] /= alphaj;
    if (npar > 1) {
        kf = 0;
        ku = 0;
        for (j = 1; j < npar; j++) {
            fj = phi[j];
            for (i = 0; i < j-1; i++) {
                kf++;
                fj += phi[i] * u[kf];
            }
            vj = fj * d[j];
            k[j] = fj;
            ajlast = alphaj;
            alphaj = ajlast + vj * fj;
            d[j] += ajlast/alphaj/lambda;
            pj = -fj / ajlast;
            for (i = 0; i < j-1; i++) {
                ku++;
                w = u[ku] + k[i] * pj;
                k[i] += u[ku] * vj;
                u[ku] = w;
            }
        }
    }
    for (i = 0; i < npar; i++) theta[i] += e * k[i] / alphaj;
}
```

References

1. Astrom, K. J. and Wittenmark, B., *Adaptive Control*, Addison Wesley, 1989.

2. Bierman, G. J., *Factorisation methods for discrete sequential estimation*, New York Acadaemic Press, 1977.