Least Squares Fitting to a Rational Transfer Function with Time Delay

Lilja, Mats

1987

Document Version:
Publisher's PDF, also known as Version of record

CODEN: LUTFD2/(TFRT-7363)/1–10/(1987)

Least Squares Fitting to a Rational Transfer Function with Time Delay

Mats Lilja

Department of Automatic Control
Lund Institute of Technology
June 1987

Department of Automatic Control
Lund Institute of Technology
P.O. Box 118
S-221 00 Lund Sweden

Document name
Report

Date of issue
June 1987

Document Number
CODEN:LUTFD2/(TFRT-7363)/1-10/(1987)

Author(s)
Mats Lilja

Supervisor
Karl Johan Åström

Sponsoring organisation

Title and subtitle
Least Squares Fitting to a Rational Transfer Function with Time Delay

Abstract

This report describes a method to fit frequency response data to a rational transfer function with a time delay. The method used is the equation error version of weighted least-squares approximation at a finite point set. This leads to a one-dimensional optimization problem which is solved by a modified Newton-Raphson method. This is implemented in the matrix manipulation language CTRL-C.

Key words
Least-squares, Approximation, Time delay

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title

ISBN

Language
English

Number of pages
10

Recipient's notes

Security classification

1. Introduction

In many applications of control theory, it is rather common that the plants to be controlled contains some time delay. This can of course be modeled by a high order, finite dimensional linear system, but sometimes it is preferrable to use a model of low order with a time delay included. This report presents a way to, from given frequency response data, separate the time delay from the low order dynamics. There is no assumptions on how the frequency response data is collected and no measurement noise is assumed. The obtained model could be used e.g. in the design of a Smith compensator but this is not pursued here. The first section makes a review of ordinary (weighted) equation error least squares fitting of points on a Nyquist curve. In the second section the 'separation method' is presented and in the third section a numerical algorithm is proposed. The algorithm is implemented in CTRL-C [CTRL-C] and some examples are given. A listing of the CTRL-C function `LSZ` used, is found in the Appendix.

2. Low order process model without time delay

Assume that the frequency response for the process is given for some frequencies

$$G(i\omega_k), \quad k = 1, 2, \ldots, N$$

G can be thought of as the transfer function of a complex model of the process. A simple process model is to be fitted to the given frequency response data. We first consider the case where the transfer function $\hat{G}$, of the simple model, is rational:

$$\hat{G}(s) = \frac{\hat{B}(s)}{\hat{A}(s)} = \frac{\hat{b}_1 s^{n-1} + \hat{b}_2 s^{n-2} + \cdots \hat{b}_n}{s^n + \hat{a}_1 s^{n-1} + \hat{a}_2 s^{n-2} + \cdots + \hat{a}_n}$$

where $2n \leq N$. The "closeness" between G and $\hat{G}$ can be measured in different ways. One choice of distance function is $\sum_{k=1}^{N} |G(i\omega_k) - \hat{G}(i\omega_k)|^2$. This gives however a problem which is non-linear in the parameters, $\hat{a}_1, \ldots, \hat{a}_n, \hat{b}_1, \ldots, \hat{b}_n$. To get linearity in the parameters, the least squares method in the equation error formulation can be utilized, that is: Find the parameter vector $\theta = (\hat{a}_1, \ldots, \hat{a}_n, \hat{b}_1, \ldots, \hat{b}_n)^T$, that minimizes the loss function

$$J(\theta) = \sum_{k=1}^{N} |\hat{A}(i\omega_k)G(i\omega_k) - \hat{B}(i\omega_k)|^2$$

One drawback with this method is that compared to the "ordinary" least squares problem, there will be a heavy weighting for high frequencies (multiplication by $|\hat{A}(i\omega_k)|$). This will deteriorate the approximation at low frequencies. To compensate for this, it is natural to introduce weighting (see next subsection).

For convenience we introduce the following notation

$$\Gamma = \mathrm{diag}(\{G(i\omega_k)\}_{k=1}^{N}) \qquad \Omega = \begin{pmatrix} (i\omega_1)^{n-1} & (i\omega_1)^{n-2} & \cdots & 1 \\ \vdots & \vdots & & \vdots \\ (i\omega_N)^{n-1} & (i\omega_N)^{n-2} & \cdots & 1 \end{pmatrix}$$

$$\phi = \Gamma \begin{pmatrix} (i\omega_1)^n \\ \vdots \\ (i\omega_N)^n \end{pmatrix} \qquad \Phi = \begin{pmatrix} -\Gamma\Omega & \Omega \end{pmatrix}$$

The loss function can now be written as

$$J(\theta) = (\Phi\theta - \phi)^*(\Phi\theta - \phi)$$

where $*$ denotes the conjugate transpose. As is well known, the explicit solution to this problem is

$$\hat{\theta} = (\Phi^*\Phi)^{-1}\Phi^*\phi$$

Computationally, however, it is more preferrable to use Cholesky factorization or singular value decomposition.

Introducing weighting

A drawback with the equation error method is, as mentioned earlier, that it puts heavy weighting on large frequencies. To get a more uniform weighting, one should introduce a weighting matrix $F = \text{diag}(\{f_k\})$ according to

$$\begin{aligned} J(\theta) &= \sum_{k=1}^{N} f_k^2 |\hat{A}(i\omega_k)G(i\omega_k) - \hat{B}(i\omega_k)|^2 \\ &= (\Phi\theta - \phi)^* F^* F(\Phi\theta - \phi) \end{aligned}$$

If $f_k = 1, \quad k = 1, 2, \ldots, N$, this corresponds to the transfer function error method

$$J(\theta) = \sum_{k=1}^{N} W(\omega_k)|G(i\omega_k) - \hat{G}(i\omega_k)|^2$$

with weighting function $W(\omega) = |\hat{A}(i\omega)|^2$ The weightings f_k should then be chosen as an apriori estimate of $|\hat{A}(i\omega_k)|^{-1}$. This could of course be done iteratively, by making several approximations with weighting equal to $|\hat{A}(i\omega_k)|^{-1}$ of the preceding approximation.

Most often one wants to have a more accurate model in certain frequency intervals (given by the control objectives), so these weightings should in turn be multiplied by further weightings, which emphasize these frequencies.

Complex frequencies

The points of approximation need not necessarily be located on the imaginary axis. An arbitrary point set $\mathcal{Z}$ in the complex plane could be chosen as the approximation set. Since all transfer functions, $G(s)$, considered here, are assumed to have the property $G(s) = \overline{G}(\overline{s})$ (equivalent to $G(s)$ having a real inverse laplace transform), the set $\mathcal{Z}$ must be closed under conjugation. This also implies that the coefficients in the approximating transfer function, will be real.

3. Low order process model with time delay

Sometimes a 'good' fitting is attained for a wider frequency range, if one introduces a time delay in the model,

$$\hat{G}(s) = \frac{\hat{B}(s)}{\hat{A}(s)}e^{-\hat{\tau}s} = \frac{\hat{b}_1 s^{n-1} + \hat{b}_2 s^{n-2} + \cdots \hat{b}_n}{s^n + \hat{a}_1 s^{n-1} + \hat{a}_2 s^{n-2} + \cdots + \hat{a}_n}e^{-\hat{\tau}s}$$

especially of course when the 'true' model has a time delay, but also in other cases (e.g. $G(s) = (s+1)^{-n}$, for large n). The Φ-matrix will in this case be dependent upon the time delay τ. Let $D(\tau)$ denote the matrix

$$D(\tau) = \mathrm{diag}(\{\exp(-i\omega_k\tau)\}_{k=1}^N)$$

The modified Φ-matrix can then be written as

$$\Phi(\tau) = \begin{pmatrix} -\Gamma\Omega & D(\tau)\Omega \end{pmatrix}$$

and the corresponding loss function

$$J(\theta,\tau) = |\Phi(\tau)\theta - \phi|^2$$

PROPOSITION. *Let P denote the matrix valued function*

$$P(\tau) = I - \Phi(\tau)(\Phi(\tau)^*\Phi(\tau))^{-1}\Phi(\tau)^*$$

Every local minimum of J with respect to τ and θ is given by

$$\hat{\theta} = \Theta(\hat{\tau})$$

where $\hat{\tau}$ is a local minimum of the function f defined by

$$f(\tau) := J(\Theta(\tau),\tau) = \phi^* P(\tau)\phi$$

and

$$\begin{aligned}\Theta(\tau) &:= \min_\theta J(\theta,\tau) \\ &= (\Phi^*(\tau)\Phi(\tau))^{-1}\Phi(\tau)^*\phi\end{aligned}$$

Proof. A necessary and sufficient condition for J to have a local minimum at $\tau = \hat{\tau}$, $\theta = \hat{\theta}$ is

$$\begin{pmatrix} \frac{\partial J}{\partial\theta} & \frac{\partial J}{\partial\tau} \end{pmatrix}_{\theta=\hat{\theta},\tau=\hat{\tau}} = 0$$

and

$$H(\hat{\theta},\hat{\tau}) := \begin{pmatrix} \frac{\partial^2 J}{\partial\theta^2} & \frac{\partial^2 J}{\partial\theta\partial\tau} \\ \frac{\partial^2 J}{\partial\theta\partial\tau} & \frac{\partial^2 J}{\partial\tau^2} \end{pmatrix}_{\theta=\hat{\theta},\tau=\hat{\tau}} > 0$$

i.e. the hessian of J with respect to (θ,τ) is positive definite at $(\hat{\theta},\hat{\tau})$. For each fixed τ, there exists a unique solution to the quadratic minimization problem in θ, given by $\Theta(\tau)$. Computing the first derivative of f, with respect to θ and τ gives

$$\begin{aligned}\frac{df}{d\tau} &= \left.\frac{\partial J}{\partial\tau}\right|_{\theta=\Theta(\tau)} + \left.\frac{\partial J}{\partial\theta}\right|_{\theta=\Theta(\tau)}\frac{d\Theta}{d\tau} \\ &= \left.\frac{\partial J}{\partial\tau}\right|_{\theta=\Theta(\tau)}\end{aligned}$$

where the last step follows from the fact that $\frac{\partial J}{\partial\theta} \equiv 0$ on the curve $(\theta,\tau) = (\Theta(\tau),\tau)$. This shows that any stationary point of J, with respect to θ and τ, corresponds to a stationary point (i.e. an extremum) of f with respect to

τ. The next step is to show that the second derivative of f with respect to τ is strictly greater than zero iff the hessian, $H(\theta,\tau)$, is positive definite. The second derivative of f is given by

$$\frac{d^2f}{d\tau^2} = \frac{\partial^2 J}{\partial\tau^2}\bigg|_{\theta=\Theta(\tau)} + 2\frac{\partial^2 J}{\partial\tau\partial\theta}\bigg|_{\theta=\Theta(\tau)}\frac{d\Theta}{d\tau} + \frac{d\Theta}{d\tau}^*\frac{\partial^2 J}{\partial\theta^2}\bigg|_{\theta=\Theta(\tau)}\frac{d\Theta}{d\tau} + \frac{\partial J}{\partial\theta}\bigg|_{\theta=\Theta(\tau)}\frac{d^2\Theta}{d\tau^2}$$

where the last term vanishes by definition. This can be rewritten as

$$\frac{d^2f}{d\tau^2} = \eta^* H(\Theta(\tau),\tau)\eta$$

where we introduced the vector η as

$$\eta(\tau) := \begin{pmatrix} \frac{d\Theta}{d\tau} \\ 1 \end{pmatrix}$$

Define the nonsingular matrix S as

$$S(\tau) := \begin{pmatrix} I & \frac{d\Theta}{d\tau}(\tau) \\ 0 & 1 \end{pmatrix}$$

Some calculations give that

$$S(\tau)^* H(\Theta(\tau),\tau)S(\tau) = \begin{pmatrix} \frac{\partial^2 J}{\partial\theta^2} & 0 \\ 0 & \frac{d^2f}{d\tau^2} \end{pmatrix}_{\theta=\Theta(\tau)}$$

where we have used the fact that $\frac{\partial J}{\partial\theta}$ is identically zero on the curve $(\theta,\tau) = (\Theta(\tau),\tau)$. This clearly implies

$$\frac{d^2f}{d\tau^2}\bigg|_{\tau=\hat{\tau}} > 0 \qquad \Longleftrightarrow \qquad H(\Theta(\hat{\tau}),\hat{\tau}) > 0$$

which was to be shown. ∎

Remark Notice that it is intuitively clear that there always exists several local minima for J. The given points can always be fitted to a Nyquist curve, which encircles the origin an arbitrarily number of times between two fitted points, by choosing a sufficiently large time delay τ. □

Remark As in the case of a process model without a time delay, some weighting should be introduced so that high frequencies are less emphasized. □

4. An algorithm

The minimization of f with respect to τ, was implemented in the matrix manipulation package CTRL-C, using a modified Newton-Raphson algorithm:

$$\tau_{n+1} = \tau_n - \frac{\frac{df}{d\tau}(\tau_n)}{\alpha\left|\frac{d^2f}{d\tau^2}(\tau_n)\right| + (1-\alpha)\frac{d^2f}{d\tau^2}(\tau_n)}$$

where $0.5 < \alpha < 1$. The modification is done in order to make local maxima repulsive and local minima attractive. Values of α less than one are chosen to get a stronger "repulsion" from a maximum. However, if the initial value of τ is chosen exactly at a maximum, the algorithm will of course get stuck. Since ϕ is independent of τ, the first and second derivatives of f are given by

$$\frac{df}{d\tau} = \phi^* \frac{dP}{d\tau}\phi$$
$$\frac{d^2f}{d\tau^2} = \phi^* \frac{d^2P}{d\tau^2}\phi$$

Recalling the definitions of P,Φ and D,

$$D(\tau) = \mathrm{diag}(\{\exp(-i\omega_k\tau)\}_{k=1}^N)$$
$$\Phi(\tau) = \begin{pmatrix} -\Gamma\Omega & D(\tau)\Omega \end{pmatrix}$$
$$P(\tau) = I - \Phi(\tau)(\Phi(\tau)^*\Phi(\tau))^{-1}\Phi(\tau)^*$$

the first derivative of P is computed according to

$$\frac{dD}{d\tau} = \mathrm{diag}(\{-i\omega_k \exp(-i\omega_k\tau)\}_{k=1}^N)$$
$$\frac{d\Phi}{d\tau} = \begin{pmatrix} 0 & \frac{dD}{d\tau}\Omega \end{pmatrix}$$
$$\frac{dP}{d\tau} = Q + Q^* \qquad Q := -P\frac{d\Phi}{d\tau}(\Phi^*\Phi)^{-1}\Phi^*$$

and the second derivative of P is given by

$$\frac{d^2D}{d\tau^2} = \mathrm{diag}(\{-\omega_k^2 \exp(-i\omega_k\tau)\}_{k=1}^N)$$
$$\frac{d^2\Phi}{d\tau^2} = \begin{pmatrix} 0 & \frac{d^2D}{d\tau^2}\Omega \end{pmatrix}$$
$$P_1 := \frac{d\Phi}{d\tau}(\Phi^*\Phi)^{-1}\Phi^*$$
$$P_2 := \frac{d^2\Phi}{d\tau^2}(\Phi^*\Phi)^{-1}\Phi^*$$
$$\frac{d^2P}{d\tau^2} = \frac{dQ}{d\tau} + \frac{dQ^*}{d\tau} \qquad \frac{dQ}{d\tau} = P(2P_1^2 - P_2) + Q^*Q + QQ^*$$

Example Consider a system with transfer function

$$G(s) = \frac{1}{(s+1)^{16}}$$

This is a typical example of a system, which behaves quite similar to a low order system with a time delay. The nyquist curve was fitted to a delayed second order model

$$\hat{G}(s) = \frac{\hat{b}_1 s + \hat{b}_2}{s^2 + \hat{a}_1 s + \hat{a}_2} e^{-\hat{\tau} s}$$

at the points

$$\mathcal{Z} = \{i0.01, i0.2, i0.4\}$$

The start value of the time delay was zero and the value reached after 6 iterations was chosen (the magnitude of the last increment was less than 0.02). This was compared with a fitting to a third order model without any time delay

$$\hat{G}(s) = \frac{\hat{b}_1 s^2 + \hat{b}_2 s + \hat{b}_3}{s^3 + \hat{a}_1 s^2 + \hat{a}_2 s + \hat{a}_3}$$

The magnitude of the error $E(i\omega) := |G(i\omega) - \hat{G}(i\omega)|$ is shown in Figure 1.

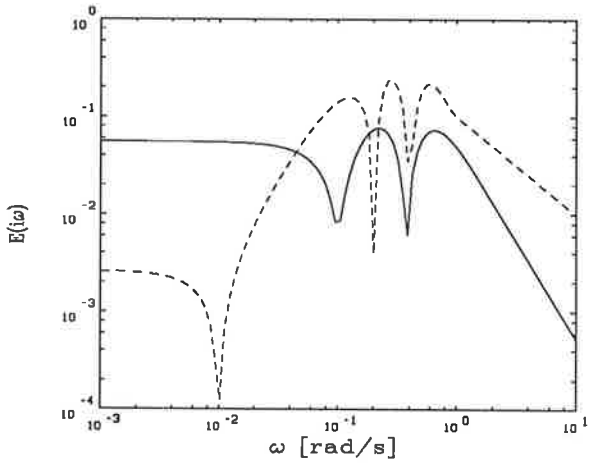

Figure 1. Magnitude of approximation errors, case 1.

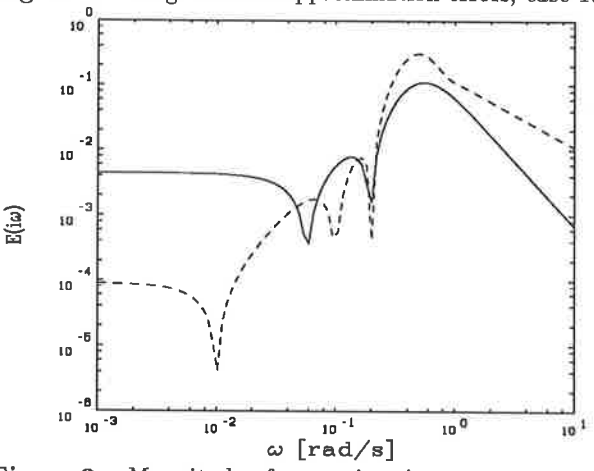

Figure 2. Magnitude of approximation errors, case 2.

The solid line corresponds to the delayed second order model and the broken line to the third order model. To illustrate the influence of the choice of frequencies, the computation was carried out for

$$\mathcal{Z} = \{i0.01, i0.1, i0.2\}$$

as well (Fig.2).

□

Notice the decrease of the error magnitude for low frequencies compared to the first choice of frequencies. Typical for pointwise approximation (especially interpolation) is the appearance of "notches" in the error magnitude curve.

5. Conclusion

A method for 'extracting' the time delay from given frequency response data has been presented. This gives a possibility to model high order dynamics by using fewer parameters (since arbitrarily large phase shifts can be modelled). The method used is an equation error version of weighted least squares fitting of a transfer function at a finite number of complex frequencies.

6. References

CTRL-C (1986): *CTRL-C User's Guide*, Systems Control Technology, Inc. Palo Alto, California, USA.

7. Appendix: Listing of the CTRL-C function LSZ

```
//[a,b,tau,ep,di,logg]=lsz(zvec,gvec,pvec,na,nb,dtmax,tau0);
//
// Finds a weighted least-squares equation error approximation
//
//                ^       - tau s  b(s)
//              G(s) = e           ----
//                                 a(s)
//
// at the complex numbers contained in the vector zvec. The
// function values are contained in the vector gvec and the
// weightings in the vector pvec. The degrees of the
// polynomials a and b are na and nb respectively. A modified
// Newton-Raphson algorithm is used to first find a locally
// minimizing tau. The iteration stops when the increment in
// tau gets less than dtmax. The start value for the iteration
// is specified by tau0. The coefficients in a and b are then
// calculated directly after inserting the obtained value of
// tau. The vector ep contains the square roots of the weighted
// error at each point in zvec while di is a vector containing
// the 'absolute' error G(z)-exp(-tau z)b(z)/a(z) at these
// points.  Finally the output variable logg is a logging of
// the iterations the values of tau and the values of the loss
// function and its first and second derivatives.
//
  nz = size(zvec)*[0;1];
  np = size(pw)*[0;1];
  nn = 2*nz;
  en = eye(nn);
  if 2*nz < na+nb+1 then disp('Too few frequencies'); return;
  s = conj([zvec conj(zvec)])';
  pw = pvec/max(pvec);
  filt = diag([pw pw]);
  g = diag([gvec conj(gvec)]);
  zj = ones(nn,1);
  psib = zj;
  for j=1:nb;..
    zj = s*.zj;..
    psib = [zj psib];..
  end;
  zj = ones(nn,1);
  psia = zj;
  for j=1:(na-1);..
    zj = s*.zj;..
    psia = [zj psia];..
  end;
  zn = s*.zj;
  gam = filt*g*zn;
  tau = tau0;                    //Starting value
```

```
dtau = 10000;
count = 0;
logg = 0;
my = 1;
while abs(dtau)>dtmax,..
  count = count + 1;..
  ew = diag(exp(-s*tau));..
  fi = filt*[-g*psia ew*psib];..
  [u,sig,v] = svd(fi);..
  u1 = u(:,1:min(size(sig)));..
  sig = sig(1:min(size(sig)),:);..
  fiff = (v/sig)*u1';..
  theta = fiff*gam;..
  tha = theta(1:na);..
  res = fi*theta - gam;..
  diff = abs(res/.(psia*tha+zn));..
  epsi = sqrt(diff'*diff/nn),..
  emax = max(diff);..
  dew = -diag(s)*ew;..
  dfi = filt*[ 0*psia dew*psib];..
  d2ew = -diag(s)*dew;..
  d2fi = filt*[ 0*psia d2ew*psib];..
  p = en - fi*fiff;..
  p1 = dfi*fiff;..
  p2 = d2fi*fiff;..
  q =  -p*p1;..
  dp = q + q';..
  dq = p*( 2*p1*p1 - p2 ) + q'*q + q*q';..
  d2p = dq + dq';..
  gpg = gam'*p*gam;..
  gdpg = gam'*dp*gam;..
  gd2pg = gam'*d2p*gam;..
  logg(count,1:4) = [tau gpg gdpg gd2pg];..
  dtau = -gdpg/(0.6*abs(gd2pg)+0.4*gd2pg),..
  tau = tau + dtau,..
  dtau;..
  tau;..
end;
// ------------------------------------------------ //
// Notice the modification in the updating of dtau:  //
// This is a fix, in order to assure convergence to a //
// local minimum, rather than to a local maximum.    //
// ------------------------------------------------ //
ew = diag(exp(-s*tau));
fi = filt*[-g*psia ew*psib];
[u,s,v] = svd(fi);
sig = s(1:min(size(s)),:);
u1 = u(:,1:min(size(s)));
fiff = (v/sig)*u1';
theta = real(fiff*gam);
tha = theta(1:na);
```

```
res = fi*theta - gam;
ep = sqrt(res*.conj(res));
ep = ep(1:nz)';
di = abs(res/.(psia*tha+zn));
di = di(1:nz)';
logg = real(logg);
emax = max(diff);
a = [1 theta(1:na)'];
b = theta(na+1:na+nb+1)';
tau = real(tau);
```