ESTIMATE PHYSICAL PARAMETERS BY BLACK-BOX MODELING

Liang-Liang Xie [*,1] and **Lennart Ljung** [**]

[*] *Institute of Systems Science, Chinese Academy of Sciences, 100080, Beijing, China*
[**] *Department of Electrical Engineering, Linköping University, 581 83, Linköping, Sweden*

Abstract: Estimating unaccessible physical parameters of a system with the information provided by the input-output data is of practical significance. This paper presents one such method based on the state-space model non-singular transformation equivalence. An algorithm is provided for solving the corresponding bilinear equations, with various modifications in special cases. Many practical issues are addressed.

Keyword: system identification, physical modeling, state-space model, bilinear equation, least square estimation.

1. INTRODUCTION

System identification is an art of modeling. Its basic philosophy is to find a good mathematical model to fit the input-output data of a system instead of building a model by physical laws, although the physical structures do provide some guidance in choosing model structures and experiment designs. The motivation comes directly from practice: First, real systems are often too complicated to model with physical laws, which govern the functioning of so many small parts and their interactions; Second, often in practice, for many purposes, e.g., control, we would be satisfied with a mathematical model that describes the input-output dynamics quite well, without asking, e.g., what is the real meaning of the parameters in the model. Whether it is possible to model the input-output dynamics of a system only by its input-output data (black-box modeling) and how to do it are the main topics of system identification. For a self-contained introduction and comprehensive covering of various methods and analyses, see (Ljung, 1999) and the references therein. For practice and real implementations, there is a System Identification Toolbox in Matlab (Ljung, 1995).

However, sometimes in practice, it is possible to do physical modeling for relatively simple systems, although there may be some uncertainties, e.g., the values of some parameters are not accessible. At the same time, we can construct a black-box model based on the input-output data by the methods in system identification. The equivalence of these two models lies in that they describe the same input-output dynamics. Now the question is if it is possible to estimate the uncertain parameters in the physical model, whose values have practical meaning.

2. PRELIMINARY DISCUSSIONS

In physical modeling, sometimes we can get a state-space model like

$$\begin{cases} \dot{x} = A_0(\theta)x + B_0(\theta)u \\ y = C_0(\theta)x + D_0(\theta)u \end{cases} \tag{1}$$

with some uncertain parameters θ in A_0, B_0, C_0 or D_0 due to the difficulty of assessing the values of them in practice.

[1] Corresponding author: <xie@control.iss.ac.cn>. Part of his work was conducted while visiting Linköping University, Sweden.

While using the input and output data $\{u(t)\}$ and $\{y(t)\}$, by the methods in system identification, we can get a black-box model like (In the case where the data are discrete, a discrete-continuous model transformation is needed, see,e.g., (Åström and Wittenmark, 1984, Chapter 3))

$$\begin{cases} \dot{x}' = Ax' + Bu \\ y = Cx' + Du \end{cases} \tag{2}$$

which can describe the same input-output dynamics as (1). The difference is that x' may not be the physical states and that A, B, C and D may not have practical meaning. But according to the linear system theory, if the system is both controllable and observable, there exists some transformation matrix T such that

$$\begin{cases} T \cdot A = A_0(\theta) \cdot T \\ T \cdot B = B_0(\theta) \\ \quad C = C_0(\theta) \cdot T \end{cases} \tag{3}$$

and $D = D_0(\theta)$. This relationship is practically useful. For example, we may use it to find out the unknown parameters θ in A_0, B_0, C_0 or D_0, which have practical meaning.

The purpose of this paper is to present an efficient algorithm to solve the set of bilinear equations (3), where both T and some parameters θ inside A_0, B_0, C_0 are unknown.

This problem is more complex than it appears. First, because of the unknown parameters θ inside A_0, B_0, C_0, the solution of T is generally not unique. Second, more unknown parameters inside A_0, B_0, C_0 would more likely lead to multiple solutions of these parameters. Actually, by examining a few simple examples, it was found that a moderately large number of unknown parameters is enough to cause several solutions. Of course, to find out the real practical values, the algorithm needs to provide all the possible solutions.

Generally, to solve equations, the less variables, the better. One way to reduce the number of variables in (3) is to convert it into a set of polynomial equations, e.g., the equivalent transfer function equations:

$$\begin{aligned} & C_0(\theta)[sI - A_0(\theta)]^{-1} B_0(\theta) + D_0(\theta) \\ & = C(sI - A)^{-1} B + D, \end{aligned} \tag{4}$$

where T disappears. There exist many ways to solve polynomial equations (see e.g. the references in (Chesi *et al.*, 2000)). Most of them are based on algebraic geometry and homotopy methods (e.g. "Gröbner bases", the method used in "Maple"). A new approach was recently proposed in (Chesi *et al.*, 2000), where LMI was employed as a medium step. Basically the method is first to introduce some virtual variables like: (for the case that there are three variables in (4): θ_1, θ_2, θ_3)

$$\begin{aligned} \theta_4 &:= \theta_1\theta_2, \quad \theta_5 := \theta_2\theta_3, \\ \theta_6 &:= \theta_1\theta_3, \quad \theta_7 := \theta_1\theta_2\theta_3. \end{aligned} \tag{5}$$

Although more variables were introduced, the benefit is that nonlinear equations are converted into linear ones. Solving a set of linear equations is trivial. The next step is to find out any rule among the solutions based on the relations (5) by observation in order to determine what to do next. As a theory, this method fits the problem (4) well. But as well known from combinatorics, the introduction of virtual variables like (5) means $2^n - 1$ variables instead of n true ones. To search for any rule among $2^n - 1$ solutions by pure observation is normally infeasible even for moderately large n.

Another big issue is that in practice (3) or (4) is always an approximation. There are always numerical errors since A, B, C came from black-box modeling. Hence some exact computation softwares such as "Maple" are not applicable. "Maple" can sometimes fail to provide any solutions even if there is. Moreover, it takes too long time (e.g. several hours or more) even when the number of unknown parameters is not large (e.g. five parameters). The computation time is exponentially increasing as the number of unknown parameters increases.

Therefore, in this study we take a numerical approach. The disadvantage with numerical methods is that (3) or (4) is a non-convex problem. Hence, the initial values have to be chosen carefully. But since A_0, B_0, C_0 came from practical physical modeling, it is realistic that we have some *a priori* knowledge on the unknown parameters θ, e.g., the lower and upper bounds. Then we only need to try different initial values between these bounds. This is one of our basic assumptions in this paper.

Generally computation time is a measure to judge an algorithm. To fully exploit the property of this problem, we starts with (3) instead of (4) although (4) contains less variables (namely, T disappears). The advantage with (3) is that it is bilinear. Hence linear least-square (LS) methods can be used (fix $\widehat{T}$ or $\widehat{\theta}$ at each time in turn), which is much faster compared with nonlinear methods such as Gauss-Newton methods.

Before we design the algorithm, a criterion must be decided. The most natural way of course is to minimize the errors of the equations (3). For example, we can choose to minimize the Frobenious Norm of the differences:

$$e = \min_{\theta,T}(\|T\cdot A - A_0(\theta)\cdot T\|_F + \|T\cdot B - B_0(\theta)\|_F + \|C - C_0(\theta)\cdot T\|_F), \quad (6)$$

where $\|\cdot\|_F$ is the Frobenious Norm: $\|M\|_F = \mathrm{tr}(M\cdot M^T)$, for any matrix M.

(6) is the criterion we will adopt for the algorithm. As can be seen from next section, it is well suited for applying linear LS method. The deficiency with (6) is that it doesn't have any real physical meaning. All our confidence is based on that there exists such a mathematical relation as (3). However, because of the complexity of the real problems (e.g., the system may not be well identifiable due to nearly zero-pole cancellations although still controllable and observable), we may come up with a black-box model which describes similar input-output dynamics (e.g., similar Bode plots), but fails to preserve the relationship (3). In such cases, it is of course wiser to choose some other criterion that is directly related to the input-output dynamics, such as the Bode plots or frequency functions. But normally such criterions are dependent on the unknown parameters in a complicated nonlinear way. Whether good algorithms can be found for such criterions is for further research.

In next section, our algorithm is presented, together with a few versions adapted to some special cases.

3. THE ALGORITHM

Our algorithm can be divided into four steps.

Step 1. Choose initial values θ_0 for the unknown parameters θ inside A_0, B_0, C_0.

Step 2. For the initial values chosen in Step 1, minimize the criterion (6) by solving for T using the linear LS method:

$$\widehat{T} = \operatorname*{argmin}_{T}(\|T\cdot A - A_0(\theta_0)\cdot T\|_F + \|T\cdot B - B_0(\theta_0)\|_F + \|C - C_0(\theta_0)\cdot T\|_F).$$

Step 3. For $\widehat{T}$ got in Step 2, minimize the criterion (6) by solving for θ using the linear LS method:

$$\hat{\theta} = \operatorname*{argmin}_{\theta}(\|\widehat{T}\cdot A - A_0(\theta)\cdot \widehat{T}\|_F + \|\widehat{T}\cdot B - B_0(\theta)\|_F + \|C - C_0(\theta)\cdot \widehat{T}\|_F).$$

Step 4. Check the value of the error

$$e_1 = (\|\widehat{T}\cdot A - A_0(\hat{\theta})\cdot \widehat{T}\|_F + \|\widehat{T}\cdot B - B_0(\hat{\theta})\|_F + \|C - C_0(\hat{\theta})\cdot \widehat{T}\|_F).$$

If e_1 is less than some confidence error bound ε_0, then accept $\hat{\theta}$ as one solution of the unknown parameters and $\widehat{T}$ as the corresponding transformation; If not, decide θ_0 is a good initial value or not: No, return to Step 1; Otherwise, let $\theta_0 = \hat{\theta}$ and return to Step 2.

3.1 *Analysis*

It is obvious that the value of the criterion (6) is always decreasing in Step 2 and Step 3. As long as in Step 1 the initial values θ_0 are well chosen, it finally will decrease to zero (if equations (3) are exact), which ensures that $\hat{\theta}$ and $\widehat{T}$ converge to one of the possible solutions. Due to numerical errors and the fact that the algorithm cannot go on infinitely, some confidence error bound ε_0 was introduced. It can be expected that this algorithm works quickly since only linear LS method is involved.

Undoubtedly, the key issue with this algorithm lies in Step 1: How to choose the initial values θ_0? Since this is a non-convex problem, any badly chosen initial values will not lead to any solutions. Anyway this is an essential issue with any numerical algorithms trying to solve non-convex problems. Fortunately, in this problem the unknown parameters θ came from practical physical modeling, on which we normally have some *a priori* knowledge. In the following, as an example, we suppose that their lower and upper bounds are known.

Example 1. Suppose $\underline{\theta} \le \theta \le \overline{\theta}$. That is to say, each element of θ lies on the interval confined by the corresponding lower and upper bounds. We may choose any value on the interval to be its initial value. But we cannot be sure that any initial values on the intervals will lead to one solution or the same solution, due to, e.g., some of these intervals may be too long. One of the obvious ways to overcome this is to divide a long interval into several short ones and on each of them try an initial value. What we can expect is that by continuing such interval refining, finally we will get all the solutions. In practice, normally the implementer should have the confidence on how small intervals are enough for all the solutions. The main drawback with this method is that the number of such chosen initial values is exponentially increasing as the number of parameters increases. This however seems inevitable for this problem.

Apparently it is important to realize whether an initial value is good or not as soon as possible. Earlier quit of bad initial values means less computation time. The decreasing rate of the value of the criterion (6) can provide some useful information in this aspect. But this is an issue highly dependent on the actual problem.

3.2 *Some special versions*

Due to the variety of the practical problems, different adaptations may be made on the general form of the algorithm. In some special cases, some modifications will greatly reduce the computation time.

Case 1. Relatively many unknown parameters lie on the same row in A_0 or C_0.

In this case, it may be wise to omit the equations introduced by such rows in (3). Let us be specific. Suppose the dimensions of A_0, B_0 and C_0 are $n \times n$, $n \times m$ and $r \times n$ respectively. It is easy to see that there are all together $n^2 + nm + nr$ equations in (3), while normally the unknown parameters inside A_0, B_0 and C_0 are much less. Hence, although each row of A_0 or C_0 will introduce n equations and the number of unknown parameters introduced is absolutely no more than n, it is still wise to drop out some rows with relatively many unknown parameters. This observation becomes vivid by considering an extreme case. Suppose all the unknown parameters are on the same row in A_0. Omitting this row from (3), we still get $n(n-1) + nm + nr$ equations, which are sufficient to solve for the n^2 variables in T as long as non-singularity is satisfied. Once T obtained, which of course is the true solution in this case, picking up the omitted equations, we can directly get the solution for the unknown parameters in that row.

Case 2. Different unknown numbers have quite different lengths of ranges.

This consideration follows Example 1. In Example 1, it is clear that longer distance between lower and upper bounds means more divisions of the interval, which in turn implies more choices of initial values, hence longer computation time. Therefore, the equations containing the unknown parameters that have wide ranges may be preferably omitted in the spirit of Case 1.

Case 3. Weighted LS methods

This may be better regarded as a general consideration instead of a special case. The basic idea is quite simple: We may have different confidences on the $n^2 + nm + nr$ equations in (3), e.g., some equations contain more unknown parameters than others. Therefore, different weights may be applied when using the LS method in Step 2 and Step 3.

4. A NUMERICAL EXAMPLE

In this section, we show a simple example to illustrate the algorithm.

We adopt the example depicted in Figure 3-15 in (Chen, 1984): A cart with an inverted pendulum hinged on top of it, which is redrawn here in Figure 1. The problem is to maintain the pendulum at the vertical position by exerting the force u.

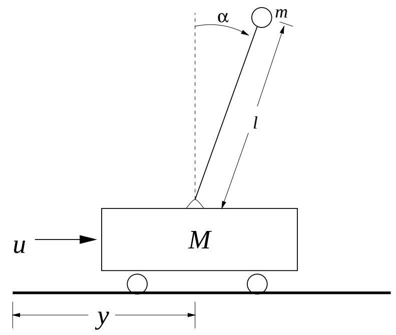

Fig. 1. A cart with an inverted pendulum.

The physical modeling is done in (Chen, 1984) and gives the following state-space model:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & \dfrac{-2mg}{2M+m} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & \dfrac{2g(M+m)}{(2M+m)l} & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0 \\ \dfrac{2}{2M+m} \\ 0 \\ \dfrac{-1}{(2M+m)l} \end{bmatrix} u \qquad (7)$$

$$y = [1 \quad 0 \quad 0 \quad 0]\, x$$

where $x_1 = y$, $x_2 = \dot{y}$, $x_3 = \alpha$, $x_4 = \dot{\alpha}$; m and M are the mass of the pendulum and the cart respectively; l is the length of the pendulum, and g is the gravitation constant 9.8 (for simplicity, we omit all the units).

Suppose from some observation data of u and y, by black-box modeling we obtain the following state-space model:

$$\begin{bmatrix} \dot{x}'_1 \\ \dot{x}'_2 \\ \dot{x}'_3 \\ \dot{x}'_4 \end{bmatrix} = \begin{bmatrix} -18.4268 & 30.1636 & -15.0469 & 3.3101 \\ -3.3101 & -5.0311 & 9.9923 & -1.6512 \\ 1.6512 & -9.9923 & 5.0311 & 3.3101 \\ -3.3101 & 15.0469 & -30.1636 & 18.4268 \end{bmatrix} \times \begin{bmatrix} x'_1 \\ x'_2 \\ x'_3 \\ x'_4 \end{bmatrix} + \begin{bmatrix} -2.5304 \\ 0.8296 \\ -0.8214 \\ 2.4391 \end{bmatrix} u$$

$$y = 10^{-3} \times [0.8118 \quad -0.8301 \quad -0.8800 \quad 0.8282]\, x'.$$

(Actually here, the above model is obtained by simulation of the system (7) with $m = 1.2$, $M = 5.5$,

$l = 2.3$. It is stabilized by a zero-order-hold linear state-feedback controller with sampling period 0.1 and white Gaussian noise is added to the control signal for two purposes: accounting for the observation errors and acting as excitation signals for identification.)

Now we try to estimate m, M and l by the equivalence of the above two state-space models. (Although it seems that it is quite feasible in practice to measure them directly, we are showing the idea here.)

Let $\theta = [\theta_1, \theta_2, \theta_3, \theta_4]^T$ with

$$\theta_1 := \frac{-2mg}{2M+m}, \quad \theta_2 := \frac{2g(M+m)}{(2M+m)l},$$
$$\theta_3 := \frac{2}{2M+m}, \quad \theta_4 := \frac{-1}{(2M+m)l}.$$

Choose the initial estimates

$$\hat{m}(0) = 1, \quad \hat{M}(0) = 7, \quad \hat{l}(0) = 2.$$

Then correspondingly,

$$\hat{\theta}(0) = [-1.3067,\, 5.2267,\, 0.1333,\, -0.0333]^T.$$

Apply the algorithm in Section 3 with 20 iterations, we will have

$$\hat{\theta}(20) = [-2.0630,\, 5.1955,\, 0.1639,\, -0.0334]^T,$$

with $e_1 = 4.7254 \times 10^{-6}$.

By the structure of the system (7) and noting the relation (4), we solve the following three equations:

$$\begin{cases} \dfrac{2g(M+m)}{(2M+m)l} = \hat{\theta}_2(20) \\ \dfrac{2}{2M+m} = \hat{\theta}_3(20) \\ \dfrac{-2g}{(2M+m)l} = \hat{\theta}_1(20)\,\hat{\theta}_4(20) - \hat{\theta}_2(20)\,\hat{\theta}_3(20) \end{cases}$$

and get

$$\hat{m}(20) = 1.0730,\ \hat{M}(20) = 5.5635,\ \hat{l}(20) = 2.0521.$$

5. REFERENCES

Åström, K.J. and B. Wittenmark (1984). *Computer-Controlled Systems - Theory and Design.* Prentice-Hall. N.J.

Chen, C.-T. (1984). *Linear System Theory and Design.* CBS College Publishing.

Chesi, G., A. Garulli, A. Tesi and A. Vicino (2000). An LMI-based approach for characterizing the solution set of polynomial systems. In: *Proc. 39th IEEE Conference on Decision and Control.* pp. 1501–1506.

Ljung, L. (1995). *The System Identification Toolbox: The Manual.* 4th ed.. The MathWorks Inc.. Natick, MA.

Ljung, L. (1999). *System Identification - Theory for the User.* 2nd ed.. Prentice-Hall. Upper Saddle River, N.J.