What can regularization offer for estimation of dynamical systems?

Lennart Ljung Tianshi Chen

Division of Automatic Control, Department of Electrical Engineering,
Linköping University, SE-581 83
Linköping, Sweden

.

Abstract: Estimation of unknown dynamics is what system identification is about and a core problem in adaptive control and adaptive signal processing. It has long been known that regularization can be quite beneficial for general inverse problems of which system identification is an example. But only recently, partly under the influence of machine learning, the use of well tuned regularization for estimating linear dynamical systems has been investigated more seriously. In this presentation we review these new results and discuss what they may mean for the theory and practice of dynamical model estimation in general.

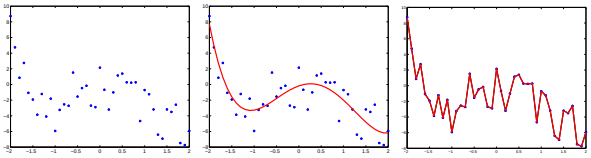

Fig. 1. Fit models but do not overfit!

1. MEET YOUR DATA WITH A PREJUDICE!

Estimation is about information in data. It is the question of squeezing out all relevant information in the data. ... but not more. In addition to relevant information, all measured data contain also irrelevant information - misinformation. In engineering we talk about signal and noise. To handle the information without getting fooled by the misinformation it is necessary to meet the data with some kind of prejudice.

Noise – Model Structure A typical prejudice that is used when building a model from data from a system, is that "nature is simple": it should be possible to describe the system with model with some simple "structure", it should belong to a set of models with restricted complexity, or has smooth responses in some sense. This should put a restriction on how much unstructured noise may affect the model.

Variance – Bias It is important to realize that the error in an estimated model has two sources: (1) We may have used too much constraints and restrictions; "too simple model sets". This gives rise to a *bias error* or *systematic error*. (2) The data is corrupted by noise, which gives rise to a *variance error* or *random error*.

In a formal, statistical sense this can be expressed as

$$\text{Mean Square Error(MSE)} = (\text{Bias})^2 + \text{Variance} \qquad (1)$$

⋆ Support from the European Research Council under the advanced grant LEARN, contract 267381 is gratefully acknowledged.

To minimize the MSE is a trade off in constraining the model: A flexible model gives small bias (easier to describe complex behaviour) and large variance (with a flexible model it is easier to get fooled by the noise), and vice versa.

This trade-off is at the heart of all estimation problems.

Data Fit – Regularization So, we should keep a keen eye on both how well the model is capable to reproduce the measured data and the complexity of the model. Conceptually, a reasonable estimation criterion is

$$\hat{\text{Model}} = \underset{\text{Model Class}}{\arg\min} \ [\text{Fit to Data} + \text{Penalty on Flexibility}] \qquad (2)$$

This codifies the basic elements in estimation. A very common way to handle the flexibility constraint is to simply restrict the model class. If an explicit penalty is added, this is known as *regularization*.

2. A FORMAL CRITERION

We shall here give a more specific version of (2): A model structure $\mathcal{M}$ is a parameterized collection of models that describe the relations between the input and output signal of the system. The parameters are denoted by θ so $\mathcal{M}(\theta)$ is a particular model. The model set then is

$$\mathcal{M} = \{\mathcal{M}(\theta) | \theta \in D_{\mathcal{M}}\} \qquad (3)$$

That model gives a rule to predict (one-step-ahead) the output at time t, i.e. $y(t)$, based on observations of previous input-output data up to time $t-1$ (denoted by Z^{t-1}).

$$\hat{y}(t|\theta) = g(t, \theta, Z^{t-1}) \qquad (4)$$

It is natural to compare the model predicted values (4) with the actual outputs and form the prediction errors

$$\varepsilon(t, \theta) = y(t) - \hat{y}(t|\theta) \qquad (5)$$

and to introduce a criterion of fit

$$F_N(\theta) = \sum_{t=1}^{N} [\varepsilon(t,\theta)^T \Lambda^{-1} \varepsilon(t,\theta)] \tag{6}$$

where Λ is a psd matrix, weighting together the different output components (channels). To add the flexibility penalty in (2) we could add a quadratic norm:

$$V_N(\theta) = F_N(\theta) + \lambda(\theta - \theta^*)^T R(\theta - \theta^*) \tag{7}$$

(λ is a scaling and R is a psd matrix). The estimate is then determined as

$$\hat{\theta}_N = \arg\min_{\theta \in D_{\mathcal{M}}} V_N(\theta) \tag{8}$$

This criterion is a clear cut balance between model fit and a penalty on the model parameter size. The amount of penalty is governed by λ and R. Clearly the model structure itself means an important model flexibility constraint.

2.1 A Maximum Likelihood View

Assume that the innovations in the system are Gaussian with zero mean and (known) covariance matrix Λ, so that

$$y(t) = \hat{y}(t|\theta) + e(t), \quad e(t) \in N(0, \Lambda) \tag{9}$$

for the θ that generated the data. Then it follows that the negative logarithm of the likelihood function for estimating θ from y is

$$L(\theta) = \frac{1}{2}[F_N(\theta) + N \log\det\Lambda + N\log 2\pi] \tag{10}$$

where $F_N(\theta)$ is defined by (6). See Lemma 5.1 in Ljung [1999]. So the Maximum Likelihood model estimate (MLE) for known Λ is obtained by minimizing $F_N(\theta)$.

3. BAYESIAN VIEW

The criterion (8) makes sense in a classical estimation framework as an ad hoc modification of the MLE to deal with possible ill-conditioned minimization problems. The added quadratic term then serves as proper *regularization* of an ill-conditioned inverse problem, see, for example, Tikhonov and Arsenin [1977].

But for a richer perspective it is useful to invoke a Bayesian view. Then the sought parameter vector θ is itself a random vector with a certain probability density function (pdf). This random vector will of course be correlated with the observations y. If we assume that the *prior distribution* of θ (before y has been observed) is Gaussian with mean θ^* and covariance matrix Π,

$$\theta \in N(\theta^*, \Pi) \tag{11}$$

its prior pdf is

$$f(\theta) = \frac{1}{\sqrt{(2\pi)^d \det(\Pi)}} e^{-(\theta-\theta^*)^T \Pi^{-1}(\theta-\theta^*)/2} \tag{12}$$

The posterior (after y has been measured) pdf then is by Bayes rule (Y denoting all measured y-signals)

$$P(\theta|Y) = \frac{P(\theta, Y)}{P(Y)} = \frac{P(Y|\theta)P(\theta)}{P(Y)} \tag{13}$$

In the last step $P(Y|\theta)$ is the likelihood function corresponding to $L(\theta)$, $P(\theta)$ is the prior pdf (12) and $P(Y)$ is a θ-independent normalization. Apart from this normalization, and other θ-independent terms, twice the negative logarithm of (13) equals $V_N(\theta)$ in (7) with

$$\lambda R = \Pi^{-1} \tag{14}$$

That means that with (14), the regularized estimate (8) is the *Maximum A Posteriori* (MAP) Estimate.

This Bayesian interpretation of the regularized estimate also gives a clue to select the regularization quantities λ, R, θ^*.

4. SOME COMMON MODEL STRUCTURES

For linear systems, a general model structure is given by the transfer function G from input to output and the transfer function H from a white noise source e to output additive disturbances:

$$y(t) = G(q,\theta)u(t) + H(q,\theta)e(t) \tag{15a}$$

$$\mathcal{E}e^2(t) = \lambda; \quad \mathcal{E}e(t)e(k) = 0 \text{ if } k \neq t \tag{15b}$$

where $\mathcal{E}$ denotes mathematical expectation. This model is in discrete time and q denotes the shift operator $qy(t) = y(t+1)$. We assume for simplicity that the sampling interval is one time unit. For normalization reasons, the function H is supposed to be *monic*, i.e. its expansion starts with a unity. The expansion of $G(q,\theta)$ in the inverse (backwards) shift operator gives the *impulse response* (IR) of the system:

$$G(q,\theta) = \sum_{k=1}^{\infty} g_k(\theta) q^{-k} u(t) = \sum_{k=1}^{\infty} g_k(\theta) u(t-k) \tag{16}$$

The natural predictor for (15) is

$$\hat{y}(t|\theta) = H^{-1}(q,\theta)[H(q,\theta) - I]y(t) + H^{-1}(q,\theta)G(q,\theta)u(t) \tag{17}$$

Since the expansion of H starts with I, the first term within brackets starts with $h_1 q^{-1}$ so there is a delay in y. The question now is how to parameterize G and H.

Black-Box Input-Output Models We now specialize to SISO (single-input single-output) models. Common *black box* (i.e. no physical insight or interpretation) parameterizations are to let G and H be rational in the shift operator:

$$G(q,\theta) = \frac{B(q)}{F(q)}; \quad H(q,\theta) = \frac{C(q)}{D(q)} \tag{18a}$$

$$B(q) = b_1 q^{-1} + b_2 q^{-2} + \ldots b_{nb} q^{-nb} \tag{18b}$$

$$F(q) = 1 + f_1 q^{-1} + \ldots + f_{nf} q^{nf} \tag{18c}$$

$$\theta = [b_1, b_2, \ldots, f_{nf}] \tag{18d}$$

C and D are, like F, monic.

A very common case is that $F = D = A$ and $C = 1$ which gives the *ARX-model*:

$$y(t) = \frac{B(q)}{A(q)}u(t) + \frac{1}{A(q)}e(t) \text{ or} \tag{19a}$$

$$A(q)y(t) = B(q)u(t) + e(t) \text{ or} \tag{19b}$$

$$y(t) + a_1 y(t-1) + \ldots + a_{na} y(t-na) \tag{19c}$$

$$= b_1 u(t-1) + \ldots + b_{nb} u(t-nb) \tag{19d}$$

Other common black/box structures of this kind are FIR (Finite Impulse Response model, $F = C = D = 1$), ARMAX ($F = D = A$), and BJ (Box-Jenkins, all four polynomials different.)

Black-box State-Space Models Another general black-box structure is to use an n:th order state space model

$$x(t+1) = Ax(t) + Bu(t) + Ke(t) \quad (20a)$$
$$y(t) = Cx(t) + e(t) \quad (20b)$$

where the state-vector x is a column vector of dimension n and A, B, C, K are matrices of appropriate dimensions. The parameters θ to be estimated consists of all entries of these matrices. Due to possible changes of basis in the state-space, there are many values of θ that correspond to the same system properties. It is easy to see that (20) describes the same models as the ARMAX model with orders n for the A, B, C- polynomials. Also, if the matrix K is fixed to zero, (20) describes the same models as the OE model with orders n for the B, F- polynomials. (See Chapter 4 in Ljung [1999].)

5. TUNING THE REGULARIZATION

5.1 General Aspects

All of the material in Sections 2 – 3 is well known in statistics since a long time, even if not utilized very much in System Identification.

An important question, that has been intensely discussed recently is that of how to select the regularization variables λ, R, θ^*. The most common special case is $R = I$ (identity matrix) and $\theta^* = 0$. Then it remains only to select the scaling λ. This special case is known as *ridge regression* in statistics (at least if the underling model is a linear regression).

The most common general tool for tuning the regularization is *cross validation*. Then the data set is split into two parts, estimation and validation data. Regularized models are then estimated using the estimation data for various values of the regularization variables, and it is evaluated how well these models can reproduce the validation data. Then pick those regularization variables that give the model with the best fit to validation data.

It is attractive to be able to estimate more directly what are good regularization variables. That can be done with a twist back to a classical view of the Bayesian calculations (13)–(14): Assume that we can parameterize $\Pi(\alpha)$ and $\theta^*(\alpha)$ with a parameter vector α that may contain a good description of θ. We can then, at least conceptually compute the likelihood function for estimating α from Y. In somewhat loose notation we have:

$$P(Y|\alpha) = \int P(Y|\theta, \alpha)P(\theta|\alpha)d\theta \quad (21)$$

Here $P(Y|\theta)$ corresponds essentially to the likelihood function $F_N(\theta)$ and $P(\theta|\alpha)$ is essentially (12) (for the parameterization $\Pi(\alpha)$ and $\theta^*(\alpha)$.) The difficulty is the integration. That could be a hard problem, except when the model is a linear regression.

5.2 Linear Regression

A *Linear Regression problem* has the form

$$y(t) = \varphi^T(t)\theta + e(t) \quad (22)$$

Here y (the scalar output) and φ (the regression vector) are observed variables, e is a noise disturbance and θ is the unknown parameter vector. In general $e(t)$ is assumed to be independent of $\varphi(t)$.

It is convenient to rewrite (22) in vector form, by stacking all the elements (rows) in $y(t)$ and $\varphi^T(t)$ to form the vectors (matrices) Y and Φ and obtain

$$Y = \Phi\theta + E \quad (23)$$

Consider (23). Suppose θ is a Gaussian random vector with zero mean and covariance matrix Π, and E is a random Gaussian vector with zero mean and covariance matrix I, and Φ is a known, deterministic matrix.Then from (23) also Y will be a Gaussian random vector with zero mean and covariance matrix

$$Z(\Pi) = \Phi\Pi\Phi^T + I \quad (24)$$

(Two times) the negative logarithm of the probability density function (pdf) of the Gaussian random vector Y will thus be

$$W(Y|\Pi) = Y^T Z(\Pi)^{-1} Y + \log\det Z(\Pi) \quad (25)$$

That will also be the negative log likelihood function for estimating Π from observations Y, so it means that we have achieved the integration of (21) so the ML estimate of Π will be

$$\hat{\Pi} = \arg\min W(Y|\Pi) \quad (26)$$

We have thus lifted the problem of estimating θ to a problem where we estimate parameters (in) Π that describe the distribution of θ. Such parameters are commonly known as *hyperparameters*.

Parameterizing Π *for FIR models* Let us now return to the IR (16) and assume it is finite (FIR):

$$G(q, \theta) = \sum_{k=1}^{m} b_k u(t-k) = \varphi_u^T(t)\theta_b \quad (27)$$

where we have collected the m elements of $u(t-k)$ in $\varphi_u(t)$ and the m IR coefficients b_k in θ_b. That means that the estimation of FIR models is a linear regression problem. All that was said above about linear regressions, regularization and estimation of hyper-parameters can thus be applied to estimation of FIR models. In particular suitable choices of Π should reflect what is reasonable to assume about an IR: If the system is stable, b should decay exponentially, and if the IR is smooth, neighboring values should have a positive correlation. That means that a typical regularization matrix (prior covariance matrix) Π^b for θ_b would be matrix whose k, j element is something like

$$\Pi^b_{k,j}(\alpha) = C\min(\lambda^k, \lambda^j);\ \alpha = [C, \lambda] \quad (28)$$

where $C \geq 0$ and $0 \leq \lambda < 1$. This is one of many possible parameterizations of Π (so called *kernels*). This choice is known as the TC-kernel. The hyperparameter α can then be tuned by (26):

$$\hat{\alpha} = \arg\min W(Y|\Pi^b(\alpha)) \quad (29)$$

Efficient numerical implementation of this minimization problem is discussed in Chen and Ljung [2013] and Carli et al. [2012].

Parameterizing Π *for ARX-models* We can write the ARX-model (19) as

$$\begin{aligned} y(t) = &- a_1 y(t-1) - \ldots - a_n y(t-n) + b_1 u(t-1) + \ldots \\ &+ b_m u(t-m) = \varphi_y^T(t)\theta_a + \varphi_u^T(t)\theta_b = \varphi^T(t)\theta \end{aligned} \quad (30)$$

where φ_y and θ_a are made up from $y(t-1),\dots,y(t-n)$ and a_k in an obvious way. That means that also the ARX model is a linear regression, to which the same ideas of regularization can be applied. Eq (30) shows that the predictor consists of two IRs, one from y and one from u and similar ideas on the parameterization of the regularization matrix can be used. It is natural to partition the Π-matrix in (11) along with θ_a, θ_b and use

$$\Pi(\alpha_1,\alpha_2) = \begin{bmatrix} \Pi^a(\alpha_1) & 0 \\ 0 & \Pi^b(\alpha_2) \end{bmatrix} \tag{31}$$

with $\Pi^{a,b}(\alpha)$ as in (28).

See Chen et al. [2012c], Pillonetto and Nicolao [2010] and Pillonetto et al. [2011], Chen et al. [2012b], Chen et al. [2012a], Dinuzzo [2012] for more information around regularization of FIR and ARX models. This technique also has links to so called Gaussian Processes Regression, extensively used in machine learning, e.g. Rasmussen and Williams [2006].

A note on the general approximation capability of ARX models. The ARX model (19) or (30) is of more general interest than it may seem. It is known, Ljung and Wahlberg [1992], that any linear system (15) can be arbitrarily well approximated by an ARX model for sufficiently large orders na, nb. We will illustrate that property in Section 7.2.

6. SOFTWARE ISSUES

In the latest release of the system identification toolbox, Ljung [2013], version R2013b, regularization has been implemented as a general tool for all estimation algorithms. This is accomplished by a new field `Regularization` in all the estimation options (`arxOptions, ssestOptions`) etc. This option has subfields

```
opt.Regularization.Lambda
opt.Regularization.R
opt.Regularization.Nominal
```

corresponding to λ, R and θ^* in (7).

The tuning of the regularization parameters for FIR and ARX models, (29), (31) is accomplished by the command

```
[Lambda,R] = arxRegul(data,[na nb nk],Kernel)
```

7. EXAMPLES

The usefulness of regularization will be illustrated in this section. All code examples refer to the System Identification Toolbox, Ljung [2013].

7.1 Bias – Variance Trade-off in FIR-modelling

Consider the problem to estimate the impulse response of a linear system as an FIR model:

$$y(t) = \sum_{k=0}^{nb} g(k)u(t-k) \tag{32}$$

These are estimated by the command `m=arx(z,[0 nb 0])` The choice of order nb is a trade off between bias (large nb is requires to capture slowly decaying impulse responses

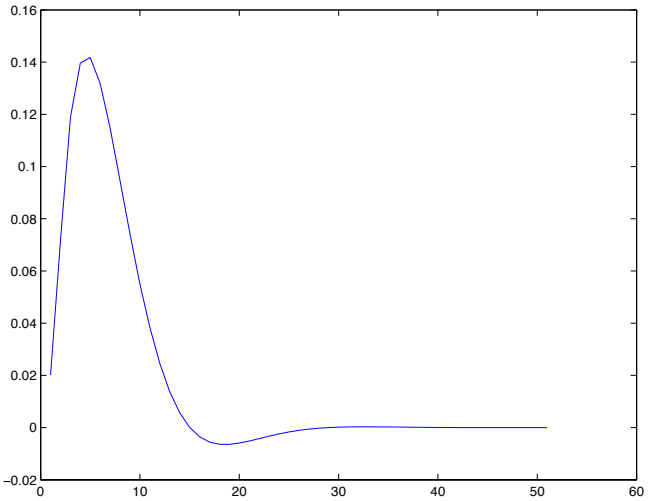

Fig. 2. The true impulse response.

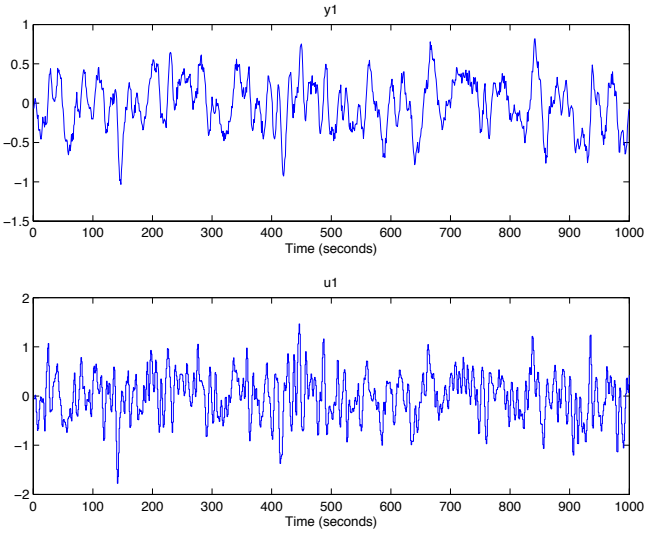

Fig. 3. The data used for estimation.

without too much error) and variance (large nb gives many parameters to estimate which gives large variance).

Let us illustrate it with a simulated example. We pick a simple second order butterworth filter as system:

$$G(z) = \frac{0.02008 + 0.04017z^{-1} + 0.02008z^{-2}}{1 - 1.561z^{-1} + 0.6414z^{-2}} \tag{33}$$

Its impulse response is shown in Figure 2. It has decayed to zero after less than 50 samples. Let us estimate it from data generated by the system. We simulate the system with low-pass filtered white noise as input and add a small white noise output disturbance with variance 0.0025 to the output. 1000 samples are collected. The data is shown in Figure 3. To determine a good value for nb we basically have to try a few values and by some validation procedure evaluate which is best. That can be done in several ways, but since we know the true system in this case, we can determine the theoretically best possible value, by trying out all models with $nb = 1,\dots,50$ and find which one has the best fit to the true impulse response. Such a test shows that $nb = 13$ gives the best error norm (mse=0.2522). This estimated impulse response is shown together with the true one in Figure 4. Despite the 1000 data points, with very

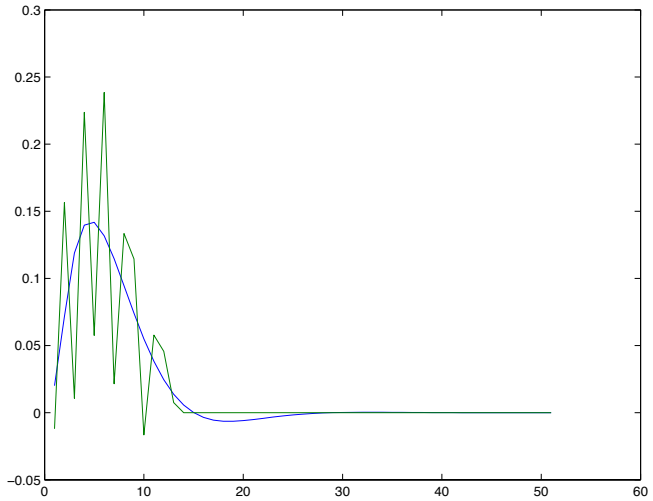

Fig. 4. The true impulse response together with the estimate for order $nb = 13$.

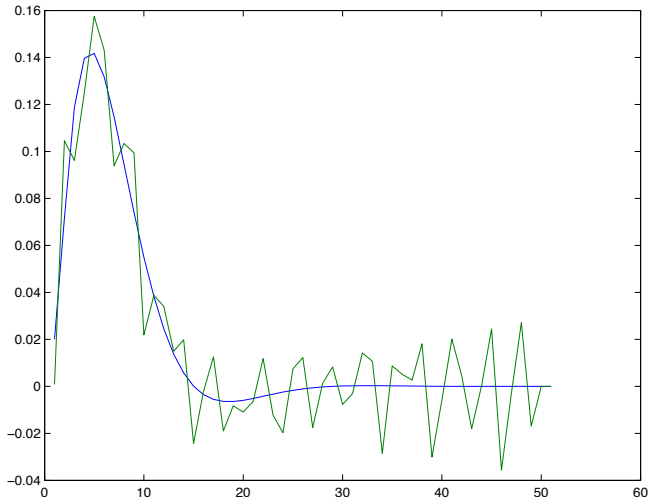

Fig. 5. The true impulse response together with the ridge-regularized estimate for order $nb = 50$.

good signal to noise ratio the estimate is not impressive. The reason is that the low pass input has poor excitation.

Let us therefore try to reach a good bias-variance trade-off by ridge regression for a FIR model of order 50:

```
aopt=arxOptions;
aopt.Regularization.Lambda=1;
m50r=arx(z,[0 50 0],aopt);
```

The resulting estimate has an error norm of 0.1171 to the true impulse response and is shown in Figure 5. Clearly even this simple choice of regularization gives a much better bias–variance tradeoff, than selecting FIR order.

We can do even better. By using the insight that the true impulse response decays to zero and is smooth, we can tailor the choice of R, λ to the data and obtain

```
[L,R]=arxRegul(z,[0 50 0],'TC');
aopt.Regularization.Lambda=L;
aopt.Regularization.R=R;
mrtc=arx(z,[0 50 0],aopt);
imtc=impulse(mrtc,50);
```

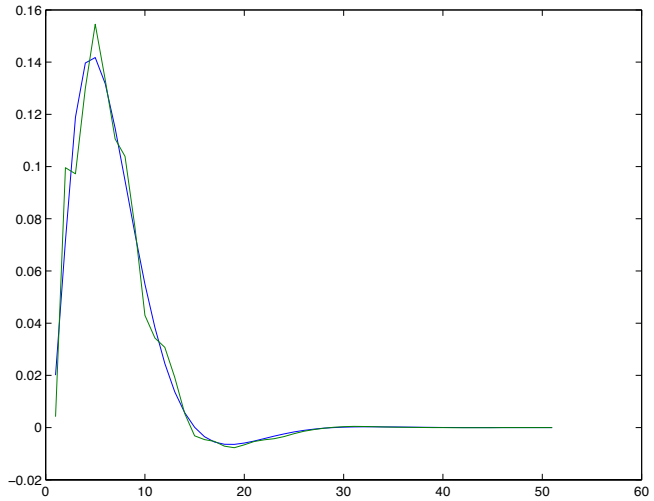

Fig. 6. The true impulse response together with the tuned regularized estimate for order $nb = 50$.

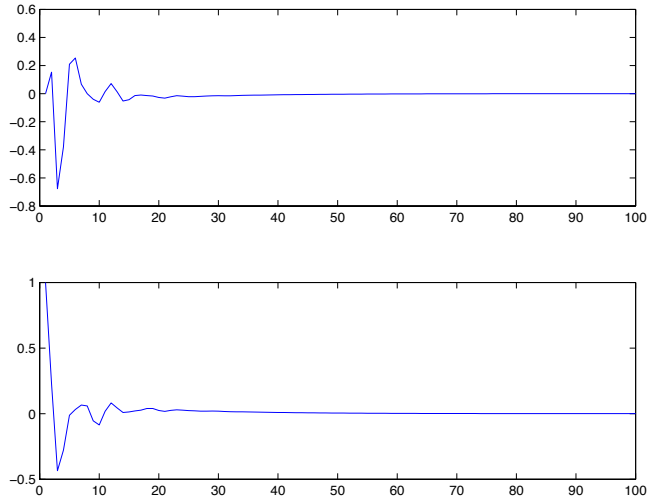

Fig. 7. The impulse responses of G (top) and H (bottom).

This gives an error norm of 0.0461 and the response is shown in Figure 6. This kind of tuned regularization is what is achieved also by the command `impulseest`.

7.2 The Use of Regularized ARX-models for Estimating State-space Models

Consider a system `m0`, which is a 30:th order linear system with coloured measurement noise:

$$y(t) = G(q)y(t) + H(q)e(t) \tag{34}$$

The impulse responses of G and H are shown in Figure 7. (`impulse(m0), impulse(noise2meas(m0))`). We have collected 210 data points `z` from (34) with a white noise input u with variance 1, and a noise level e with variance 0.1. The data is shown in Figure 8. To estimate the impulse responses of `m0` from these data, we can naturally employ state-space models of order k in innovations form,

```
mk = ssest(z,k,'ts',1);
```

(or equivalently ARMAX models), and find `impulse(mk), impulse(noise2meas(mk))`. The catch is to determine a good order `k`. There are two commonly used methods:

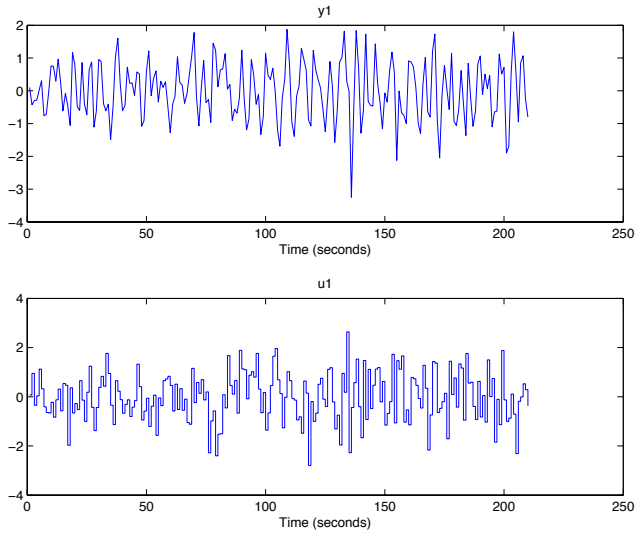

Fig. 8. The data used for estimation.

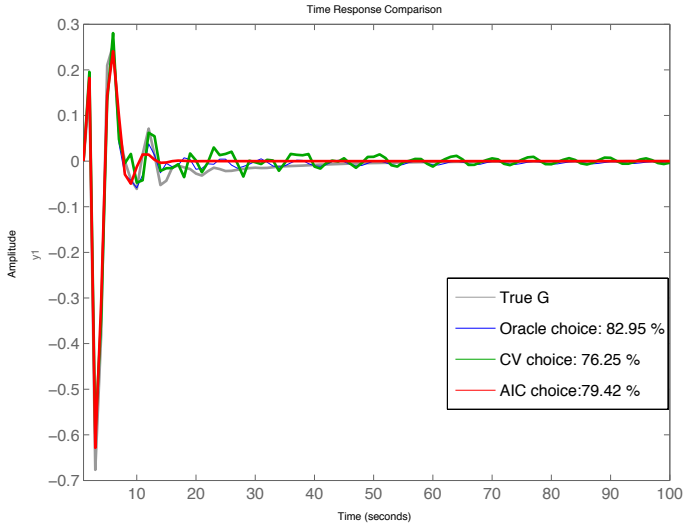

Fig. 9. The true impulse response of G compared to state space models of different orders. The figures refer to the "fit", i.e. how much (in %) of the impulse response variation in reproduced by the model. 100% thus means a perfect fit.

- *Cross validation, CV:* Estimate `mk` for $k = 1, \ldots, maxo$ using the first half of the data, `ze=z(1:150)`, and evaluate the fit to the second half of the data `zv=z(151:end)`, `[~,fitk]=compare(zv,mk,'ini','z')` and determine the order `k` that maximizes `fitk`. Then reestimate the model using the whole data record.
- *Use the Akaike criterion AIC:* Estimate models for orders $k = 1, \ldots, maxo$ using the whole data set, and then pick that model that minimizes `aic(mk)`.

Applying these techniques to the data with a maximal order $maxo = 30$ shows that cross validation picks **order 15** and AIC picks **order 3**.

There is a test that can be done ("The oracle") if we know `m0`, which of course cannot be done in practice: We check which of the models `mk` show the best fit of G resp H to the true system. For the current data, it follows that **order 12** gives the best for G and **order 3** gives the best fit for H.

In figure 9 we show how all these models compare to the true impulse response for$G$. Figure 10 shows the same plot for H.

We see that a **fit of 82.95%** is possible to achieve for G among the state-space models, but the order selection procedure may not find that best order.

We then turn to what can be obtained with regularization. Recall from Section 5.2 that a sufficiently high order ARX model can approximate a general linear system (34) arbitrarily well. We thus estimate a rather high order, regularized ARX-model by

```
aopt = arxOptions;
[Lambda,R] = arxRegul(z,[5 60 0],'TC');
aopt.Regularization.R = R;
aopt.Regularization.Lambda = Lambda;
mr = arx(z,[5 60 0],aopt);
nmr = noise2meas(mr);
```

It turns out that this regularized arx model shows a fit to the true G of 83.55% which is even better than the oracle! The fit to H is 81.50% which also is better than the oracle choice of order for best noise model.

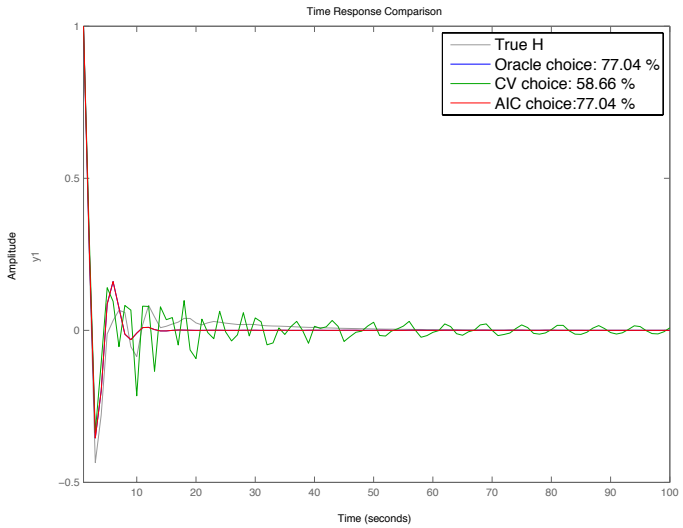

Fig. 10. The true impulse response of H compared to various models.

It could be argued that `mr` is a high order (60 states) model, and it is unfair to compare it with lower order state space models. But this high order model can be reduced to, say, order 7 by

```
mred7=balred(idss(mr),7);
nmred7=noise2meas(mred7);
```

without any essential loss of accuracy. Figures 11 and 12 shows how the regularized and reduced order regularized models compare with the oracle choice of state-space order for `ssest`.

A natural question to ask is whether the choice of orders in the ARX model is as sensitive a decision as the state space model order in `ssest`. Simple test, using e.g.

```
arx(z,[10 50 0],aopt)
```

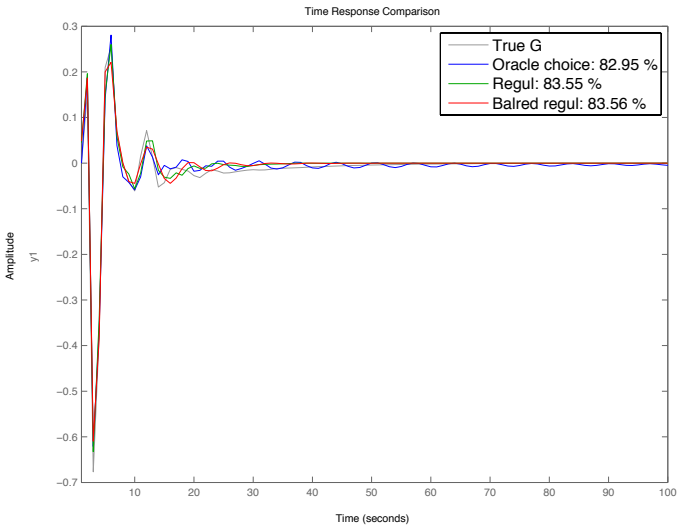

Fig. 11. The regularized models compared to the oracle choice for G.

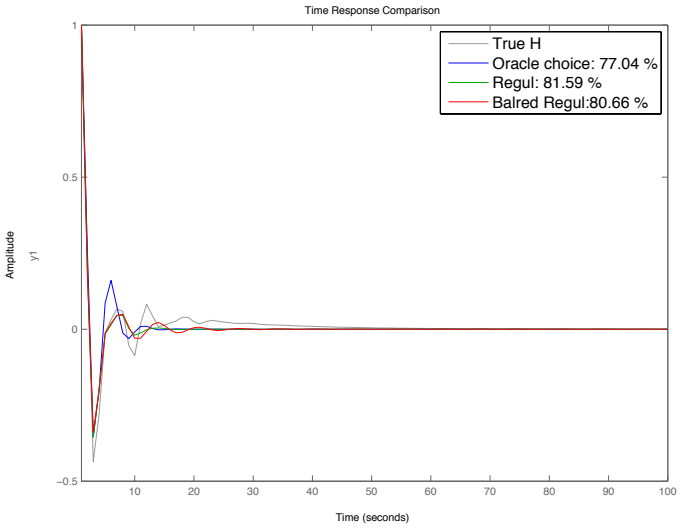

Fig. 12. The regularized models compared to the oracle choice for H.

only shows minor changes in the fit of G (82.28 % instead of 83.55 %)

8. ISSUES OF RECURSIVENESS AND ADAPTIVITY

Evidently regularization can be most useful for estimating models of dynamical systems. That means that it will be interesting to use it also in adaptive situations where the system may be changing and data information is supplied continuously. Traditionally such adaptive schemes have been engined by some sort of recursive identification technique, see e.g. Ljung [1999], Ch. 11. Let us give some comments on how regularization (of ARX models) appears in that light.

The estimate of a regularized linear regression model (22) is obtained as

$$\hat{\theta}(t) = \left(\sum_{k=1}^{t} \varphi(k)\varphi^T(k) + R \right)^{-1} \sum_{k=1}^{t} \varphi(k)y(k) \qquad (35)$$

where R is the regularization matrix in (7) (with $\theta^* = 0$). This expression can be written recursively as the well known RLS (recursive least squares) algorithm with no explicit matrix inversion. The regularization then only acts as an initial condition for the matrix update and does not affect the algorithm.

Now, if we use tuned regularization as in (28)–(29) (as implemented in `arxRegul`) it is natural to retune R as more information becomes available. If $R = R(t)$ is time varying, the recursive update of (35) becomes

$$\begin{aligned} \hat{\theta}(t) =& \hat{\theta}(t-1) + S(t)\left(\varphi(t)y(t) - \right. \\ & \left. [\varphi(t)\varphi^T(t) + R(t) - R(t-1)]\hat{\theta}(t-1)\right) \end{aligned} \qquad (36)$$

where $S(t)$ is the matrix inverse in (35).

It is worth noting that with the progress in numerical linear algebra the advantage to avoid matrix inversion for moderate sizes is less pronounced. Solving the "off-line expression" (35) by "backslash" in MATLAB on an ordinary laptop takes about 0.5 microseconds for a model with 100 parameters. That is about the same time one step of (36) takes, even without updating $S(t)$.

So the recursive update on models of moderate sizes is not a big deal. Most of the time for a tuned regularization will anyway lie in the updates of the hyperparameters (29), which is solved by a Gauss-Newton search algorithm, Chen and Ljung [2013]. To do that adaptively at time t, it is natural to form the criterion $W(Y|\alpha)$ with updated and suitably time-weighted observations, and perform just one minimization iteration starting from the current hyper-parameter estimate $\hat{\alpha}(t-1)$ to determine $\hat{\alpha}(t)$.

9. CONCLUSIONS

Regularization as such is nothing new. It has been used extensively in statistics for a long time. It has been less used in system identifiaction though. We have seen in the examples that carefully tuned regularization matrices can have a very significant effect on a good bias–variance trade-off. This was very visible in Section 7.1, where a remarkaby good estimate of the impulse response could be achieved through thoughtful regularization. It is clear that in this case the poor information in data about high frequency properties were complemented by important prior information. That information was however not very precise ("exponentially decaying and smooth"), and it was tuned to the data.

Perhaps it is more thought provoking (Section 7.2) that the 7th order balanced reduced state space model from the regularized ARX estimate is more accurate than the 7th order ML estimate of the same structure. [In fact for the data in the test, better than any finite order ML estimate!] We are used to think of MLE as "the best estimates", with optimal asymptotic properties. But for the tested data, with only 210 measurements and a complex system, we are far from the asymptotics and model quality is determined more by a well balance bias–variance trade-off. A further point to stress that the regularized estimates are less sensitive to choice of orders (since the regularization is the main constraining feature), while MLE may be very dependent on a good choice of order.

In any case, it appears to be important to complement one's toolbox of estimation tools for dynamical systems with well tuned methods for regularization.

REFERENCES

F.P. Carli, A. Chiuso, and G. Pillonetto. Efficient algorithms for large scale linear system identification using stable spline estimators. In *Proceedings of the 16th IFAC Symposium on System Identification (SysId 2012)*, 2012.

Tianshi Chen and Lennart Ljung. Implementation of algorithms for tuning parameters in regularized least squares problems in system identification. *Automatica*, 50, 2013. to appear.

Tianshi Chen, Martin S. Andersen, Lennart Ljung, Alessandro Chiuso, and Gianluigi Pillonetto. System identification via sparse multiple kernel-based regularization using sequential convex optimization techniques. *IEEE Transactions on Automatic Control, Submitted*, 2012a.

Tianshi Chen, Lennart Ljung, Martin Andersen, Alessandro Chiuso, P. Carli Francesca, and Gianluigi Pillonetto. Sparse multiple kernels for impulse response estimation with majorization minimization algorithms. In *IEEE Conference on Decision and Control*, pages 1500–1505, Hawaii, Dec 2012b.

Tianshi Chen, Henrik Ohlsson, and Lennart Ljung. On the estimation of transfer functions, regularizations and Gaussian processes-Revisited. *Automatica*, 48(8):1525–1535, 2012c.

F. Dinuzzo. Kernels for linear time invariant system identification. Manuscript, Max Planck Institute for Intelligent Systems, Spemannstrasse 38,72076 Tübingen, Germany, 2012.

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

L. Ljung. *The System Identification Toolbox: The Manual*. The MathWorks Inc. 1st edition 1986, Edition 8.3 2013, Natick, MA, USA, 2013.

L. Ljung and B. Wahlberg. Asymptotic properties of the least-squares method for estimating transfer functions and disturbance spectra. *Adv. Appl. Prob.*, 24:412–440, 1992.

G. Pillonetto and G. De Nicolao. A new kernel-based approach for linear system identification. *Automatica*, 46(1):81–93, January 2010.

G. Pillonetto, A. Chiuso, and G. De Nicolao. Prediction error identification of linear systems: a nonparametric Gaussian regression approach. *Automatica*, 47(2):291–305, 2011.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, Cambridge, MA, 2006.

A. N. Tikhonov and V. Y. Arsenin. *Solutions of Ill-posed Problems*. Winston/Wiley, Washington, D.C., 1977.