Proceedings of the
47th IEEE Conference on Decision and Control
Cancun, Mexico, Dec. 9-11, 2008

The Use of Nonnegative Garrote for Order Selection of ARX Models

Christian Lyzell, Jacob Roll, and Lennart Ljung

Division of Automatic Control
Linköpings Universitet
SE–581 83 Linköping, Sweden
{lyzell,roll,ljung}@isy.liu.se

***Abstract*— Order selection of linear regression models has been thoroughly researched in the statistical community for some time. Different shrinkage methods have been proposed, such as the Ridge and Lasso regression methods. Especially the Lasso regression has won fame because of its ability to set less important parameters exactly to zero.**

However, these methods do not take dynamical systems into account, where the regressors are ordered via the time lag. To this end, a modified variant of the nonnegative garrote method will be analyzed.

I. INTRODUCTION

Estimating the orders of ARX models is a widely studied and old topic, see e.g. [1] and [4]. One can efficiently estimate many ARX models of different orders, by building up a large data covariance matrix, and then selecting submatrices of that. If we decide upon a certain sequence in which to increase the model orders, all such models can be estimated simultaneously in an efficient manner by QR-factorization, see e.g. [5].

So what more can be said about this old problem? In this contribution we take a relatively recent statistical method for regularization, Nonnegative Garrote [3], as a starting point for a new excursion in the area.

With this method it is possible to define sequences of natural ordering of model complexity that are not one-dimensional. We may certainly say that a model with more poles than another is more complex/flexible, and similarly for zeros. But is a model with, say, 2 poles and 4 zeros more or less complex than one with say 3 poles and 2 zeros?

We define a procedure for testing all model orders, ordered independently in terms of number of zeros and in number of poles. While the obtained models are regularized, the procedure still gives insight into which parameters are the most important ones for giving a good fit to data.

The presentation is organized as follows: Section II defines the problem dealt with in this paper: II-A gives a short review of the ARX model structure, II-B a description of the original nonnegative garrote method, as presented in [3], is given. Also, a modification of the method for ARX model structure is proposed in Section II-C. Section III presents an algorithm for efficiently solving the modified nonnegative garrote problem. In Section IV the algorithm is validated on simulated ARX model data. In addition, an example on real life data collected from a "Fan and Plate" process is presented. Finally, in Section V conclusions are drawn and future work outlined.

II. PROBLEM STATEMENT

A. The ARX model structure

In this paper, we will focus on the ARX model structure [4]

$$\begin{aligned} y(t) + a_1 y(t-1) + \cdots + a_{n_a} y(t-n_a) \\ = b_1 u(t-1) + \cdots + b_{n_b} u(t-n_b) + e(t), \end{aligned} \quad (1)$$

where $u(t)$, $y(t)$ and $e(t)$ denote the input, the output and the noise signals, respectively. It is well known that (1) can be rewritten as a linear regression

$$y(t) = \varphi^T(t)\theta + e(t), \quad (2)$$

where

$$\theta = \begin{pmatrix} a_1 & \cdots & a_{n_a} & b_1 & \cdots & b_{n_b} \end{pmatrix}^T,$$
$$\varphi(t) = \begin{pmatrix} -y(t-1) & \cdots & -y(t-n_a) & u(t-1) & \cdots & u(t-n_b) \end{pmatrix}^T.$$

The FIR model structure is a special case of the ARX family when no output lag is present.

Given a dataset $Z^N \triangleq \{u(0), y(0), \ldots, u(N), y(N)\}$, the least square (LS) parameter estimate for the linear regression model (2) is then

$$\hat{\theta}_N \triangleq \arg\min_\theta V_N(\theta, Z^N), \quad (3)$$

where

$$V_N(\theta, Z^N) \triangleq \frac{1}{N} \sum_{t=1}^{N} (y(t) - \varphi^T(t)\theta)^2. \quad (4)$$

By batching the data

$$Y = \begin{pmatrix} y(1) & y(2) & \cdots & y(N) \end{pmatrix}^T, \quad (5)$$
$$\Phi = \begin{pmatrix} \varphi(1) & \varphi(2) & \cdots & \varphi(N) \end{pmatrix}^T, \quad (6)$$

978-1-4244-3124-3/08/$25.00 ©2008 IEEE

the optimal solution to (3) can be written

$$\hat{\theta}_N = \left[\Phi^T\Phi\right]^{-1}\Phi^T Y \triangleq \Phi^\dagger Y, \tag{7}$$

For more information about linear regression models and the estimation of such, see e.g. [4].

B. The Nonnegative Garrote

The Nonnegative Garrote (NNG) method was first presented in [3] as a coefficient shrinkage method for linear regression models in statistics. As the celebrated Lasso method [7], it uses regularization to penalize the size of the parameters. However, instead of affecting the parameters directly, the NNG method penalizes the least squares solution by attaching weights to it, which in turn are regularized. Thus, given the least square estimate $\hat{\theta}$ of a linear regression model (2), the NNG problem can be written as

$$\begin{aligned} \min \quad & \sum_{t=1}^{N}\left(y(t) - \sum_{j=1}^{n} w_j\varphi_j(t)\hat{\theta}_j\right)^2 + \lambda\sum_{j=1}^{n} w_j \\ \text{s.t.} \quad & w \succeq 0, \end{aligned} \tag{8a}$$

where λ is a model complexity parameter, and $\preceq$ denotes componentwise inequality. As λ increases, the weights of the less important regressors will shrink, and finally end up exactly zero. For each given λ, the NNG parameter estimate has the elements $w_i\hat{\theta}_i$, $1 \le i \le n$, where w is the optimal solution to (8a). Thus, as λ increases, the model becomes less complex.

C. Modification

In system identification, one is typically interested in the estimation of dynamical systems, compared to the static models used in statistics. In dynamic linear regression models, the regressors are naturally ordered by their time lag. The higher model order, the more saved data is needed. The original NNG method (8a) do not take such orderings into consideration. It just sets the weights of the less important regressors low, not considering their order. To be able to penalize higher order lags first, one could modify (8a) by adding some constraints on the weights. For ARX models, these constraints could be

$$1 \ge w_1 \ge w_2 \ge \cdots \ge w_{n_a} \ge 0 \tag{8b}$$

$$1 \ge w_{n_a+1} \ge w_{n_a+2} \ge \cdots \ge w_{n_a+n_b} \ge 0. \tag{8c}$$

This is a natural extension of the NNG method, for order selection of ARX models in system identification. Note that, contrary to the method in [5], we can let the ordering of y and u be independent. This yields automatic order selection, and a natural way to choose the importance between input lag and output lag, since both get an equal chance.

The modified NNG problem (8), can be rewritten as a quadratic problem with linear inequality constraints, i.e.

$$\begin{aligned} \min \quad & \frac{1}{2}w^T Q w + f^T w + \lambda\mathbf{1}^T w \\ \text{s.t.} \quad & Aw \preceq b, \end{aligned} \tag{9}$$

where $Q = 2\hat{\Theta}\Phi^T\Phi\hat{\Theta}$, $f = -2\hat{\Theta}\Phi^T Y$, $\hat{\Theta} \triangleq \operatorname{diag}(\hat{\theta})$, and the inequality constraints are derived from (8b)–(8c).

Given the solution w_λ to (9), for a specific λ, the modified NNG parameter estimate is $\hat{\theta}_\lambda = \hat{\Theta}w_\lambda$.

III. THE ALGORITHM

An efficient way to solve (9) is to use path following parametric optimization. This idea is not new and the application of parametric optimization to the original NNG problem (8a) has already been published in [9].

In the paper [6] it is shown that the problem

$$\begin{aligned} \min \quad & L(x) + \lambda J(x) \\ \text{s.t.} \quad & Ax \preceq b \\ & \bar{A}x = \bar{b}, \end{aligned} \tag{10}$$

where $L(x)$ is piecewise quadratic and $J(x)$ is a piecewise affine function, both convex, has a piecewise affine solution path, i.e. the optimal solution x to (10) is a piecewise affine function of $\lambda \in \mathbb{R}_+$. In the modified NNG problem (9), L is quadratic and J is linear as functions of w, which yields a somewhat simpler solution.

The Lagrangian to (9) is

$$\mathcal{L}(w,\mu) = \frac{1}{2}w^T Q w + f^T w + \lambda\mathbf{1}^T w + \mu^T(Aw - b), \tag{11}$$

where μ is a vector of Lagrangian multipliers. This yields the Karush-Kuhn-Tucker (KKT) conditions [2]

$$Qw + f + \lambda\mathbf{1} + A^T\mu = 0 \tag{12a}$$

$$Aw - b \preceq 0 \tag{12b}$$

$$\mu_j(A_j w - b_j) = 0 \tag{12c}$$

$$\mu \succeq 0,\ \lambda \ge 0. \tag{12d}$$

Now, let

$$\mathcal{J}^a = \{j_1, j_2, \ldots, j_{n^{\mathcal{J}}}\}, \tag{13}$$

be the set of active constraints (for μ_j). Solving (12) is then equivalent to solving

$$\begin{pmatrix} Q & A_{\mathcal{J}^a}^T \\ A_{\mathcal{J}^a} & 0 \end{pmatrix}\begin{pmatrix} w \\ \mu_{\mathcal{J}^a} \end{pmatrix} = \begin{pmatrix} -f - \lambda\mathbf{1} \\ b_{\mathcal{J}^a} \end{pmatrix}. \tag{14}$$

Differentiation with respect to λ yields

$$\begin{pmatrix} Q & A_{\mathcal{J}^a}^T \\ A_{\mathcal{J}^a} & 0 \end{pmatrix}\begin{pmatrix} \frac{\partial w}{\partial\lambda} \\ \frac{\partial\mu_{\mathcal{J}^a}}{\partial\lambda} \end{pmatrix} = \begin{pmatrix} \mathbf{1} \\ 0 \end{pmatrix}. \tag{15}$$

These equations are all that is needed to find the piecewise affine solution path. The algorithm presented in [6], tailored for the modified NNG problem, is given below in Algorithm 1.

To find the initial solution, let $\lambda = 0$. Then (9) is an ordinary least square problem with inequality constraints with the obvious solution $w = \mathbf{1}$. To find $\mu_{\mathcal{J}^a}$ we need to solve (12a). Since

$$\begin{aligned} Qw + f &= 2\hat{\Theta}\Phi^T\Phi\hat{\Theta}\mathbf{1} - 2\hat{\Theta}\Phi^T Y = 2\hat{\Theta}\Phi^T(\Phi\hat{\Theta}\mathbf{1} - Y) \\ &= 2\hat{\Theta}\Phi^T(\Phi\hat{\theta} - Y) = 0, \end{aligned} \tag{16}$$

where the last equality follows from the fact that the regression matrix Φ is orthogonal to the residuals $\Phi\hat{\theta} - Y$, one finds that $\mu_{\mathcal{J}^a} = \mathbf{0}$.

Algorithm 1 Parametric Optimization for the modified Nonnegative Garrote

1) *Initialization:* Set $\lambda = 0$, $w_0 = \mathbf{1}$, $\mu_{\mathcal{J}^a} = \mathbf{0}$ and $\mathcal{J}^a = \{1, 2, \ldots, p-1\}$. Let $S = \{(\lambda, w)\} = \{(0, \mathbf{1})\}$.
2) *Directions:* Solve (15) for $\frac{\partial w}{\partial \lambda}$ and $\frac{\partial \mu_{\mathcal{J}^a}}{\partial \lambda}$.
3) *Step length:* Find the minimal $\delta\lambda \geq 0$ satisfying one of the following:
   a) If $A_j(w_\lambda + \frac{\partial w}{\partial \lambda}\delta\lambda) = b_j$ and $A_j \frac{\partial w}{\partial \lambda} > 0$ for some $j \notin \mathcal{J}^a$. Then move the corresponding j to $\mathcal{J}^a$.
   b) If $\mu_j + \frac{\partial \mu_j}{\partial \lambda}\delta\lambda = 0$ and $\frac{\partial \mu_j}{\partial \lambda} < 0$ for some $j \in \mathcal{J}^a$. Then remove the corresponding j from $\mathcal{J}^a$.

   If no feasible solution $\delta\lambda \geq 0$ exists, set $\delta\lambda = \infty$.
4) *Update:* Set $\lambda := \lambda + \delta\lambda$, $w_\lambda := w_\lambda + \frac{\partial w}{\partial \lambda}\delta\lambda$, $\mu_{\mathcal{J}^a} := \mu_{\mathcal{J}^a} + \frac{\partial \mu_{\mathcal{J}^a}}{\partial \lambda}\delta\lambda$ and add the new solution $S := \{S, (\lambda, w)\}$.
5) *Termination criterion:* Stop if $\lambda = \infty$, else continue from step 2.

IV. Simulations

In this section, two examples where the modified NNG algorithm is applied to simulated system is presented. Additionally, a final example on real life data is given.

In the two simulation examples, the ARX(9,3) model

$$A(q)y(t) = B(q)u(t) + e(t), \tag{18a}$$

where the noise is white Gaussian with variance 0.1 and

$$\begin{aligned} A(q) = 1 &- 1.25q^{-1} + 0.4375q^{-2} - 0.3594q^{-3} \\ &+ 0.1719q^{-4} + 0.3125q^{-5} - 0.2764q^{-6} \\ &+ 0.1360q^{-7} - 0.0769q^{-8} + 0.0137q^{-9} \end{aligned} \tag{18b}$$

$$B(q) = 1 + 0.25q^{-1} - 0.25q^{-2} \tag{18c}$$

will used as the true system.

The system (18) was simulated using white Gaussian noise of unit variance as input, with $2N$ data points where $N = 1000$. The input and output were then divided equally into estimation Z_e^N and validation data Z_v^N.

To validate the different model outcomes, we will use

$$\text{fit} = 100\left(1 - \frac{\sqrt{\sum_{t=1}^{N}(y(t) - \hat{y}(t|\theta(\lambda)))^2}}{\sqrt{\sum_{t=1}^{N}(y(t) - \bar{y}))^2}}\right) \tag{19}$$

which is a measure of how much better the model describes the process compared to the mean of the output.

For the parameter estimation of the initial ARX models ($\hat{\theta}$ in (8)), the SYSTEM IDENTIFICATION TOOLBOX (SITB) in MATLAB was used.

A. Simulation: Perfect model order

Using the command `arx(Ze,[9,3,0])` in MATLAB, which is the true model structure and order (18), one gets the least square estimate of the parameter vector $\hat{\theta}$. Plugging this estimate into the NNG problem (9) yields a piecewise affine solution path w_λ and the fit values (19), calculated for validation data Z_v^N, shown in Figure 1.

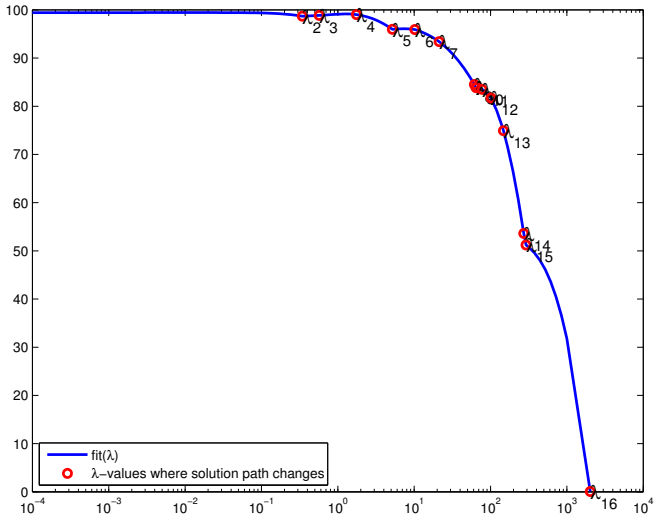

Fig. 1. The fit_λ values for the breakpoints and 1000 uniformly logarithmic distributed values of λ, calculated on validation data, for the simulation in Section IV-A.

In Table I, the parameter estimates $\hat{\theta}_\lambda$ for different values of λ ($\lambda = 0$ yields the LS estimate $\hat{\theta}$).

We see that a model reduction could be made, without to much loss in fit value, by choosing $\lambda = \lambda_6$. This is an ARX(6,3) model, with a fit value of 95.9 on validation data. This implies that the NNG method may be used as an alternative method for model reduction of ARX models.

Let us now consider a case where overfitting occurs.

B. Simulation: Overfitting

Now, by running the command `arx(Ze,[20,10,0])` in MATLAB, which is an overfitted model of (18), one gets the least square estimate $\hat{\theta}$ of the parameter vector. Plugging this estimate into the NNG problem (9) yields a piecewise affine solution path w_λ and the fit values (19), calculated for validation data Z_v^N, shown in Figure 2.

The NNG parameter estimates $\hat{\theta}_\lambda$ for some interesting breakpoints λ are given in Table II. In Figure 2, we see that there is a maximum in fit value at $\lambda = \lambda_{29}$, which corresponds to the correct model order according to Table II. Thus, it is interesting to see that the NNG method was able to find the correct model order in this case, and that the estimates for this λ value are quite near the true parameters.

In SITB there are commands (`struc`, `arxstruc`, and `selstruc`) for automatic order selection for ARX models. Running these on the same data as above yields a different solution path compared to Table II. This could be explained by the fact that once a weight has been reduced to zero by the NNG algorithm, it stays zero, while `arxstruc` has full freedom in choosing n_a and n_b just as

TABLE I

THE ESTIMATED NNG PARAMETERS IN THE SIMULATION IN SECTION IV-A FOR THE SOME OF THE INTERESTING BREAKPOINTS λ AND THEIR FIT VALUE.

λ	λ_1	λ_2	λ_5	λ_6	λ_9	λ_{12}
	0	0.34204	5.1801	10.2184	64.5728	101.4516
	-1.2625	-1.2593	-1.2343	-1.2135	-1.0506	-0.99427
	0.42517	0.41876	0.37972	0.34957	0.14967	0.1071
	-0.32125	-0.31437	-0.27963	-0.26101	-0.10834	-0.08092
	0.14716	0.14401	0.1281	0.11957	0.048297	0.037069
	0.33351	0.32411	0.2903	0.27097	0.10945	0.084008
$\hat{\theta}_\lambda$	-0.29512	-0.27663	-0.19921	-0.15468	0	0
	0.14266	0.12037	0.025542	0	0	0
	-0.080447	-0.053124	0	0	0	0
	0.01583	0	0	0	0	0
	1.0017	1.0017	1.0017	1.0017	0.99346	0.9732
	0.23916	0.23916	0.23916	0.23916	0.2372	0.23236
	-0.27722	-0.27722	-0.27044	-0.25955	-0.11118	0
fit_λ	99.45	98.68	95.97	95.91	83.90	81.72

TABLE II

THE ESTIMATED NNG PARAMETERS IN THE SIMULATION IN SECTION IV-B FOR THE SOME OF THE INTERESTING BREAKPOINTS λ AND THEIR FIT VALUE.

λ		λ_1	λ_{21}	λ_{29}	λ_{37}	λ_{40}	λ_{41}
	true	0	0.03540	0.2060	7.8785	52.5885	56.2759
	-1.2500	-1.2823	-1.2803	-1.2754	-1.2389	-1.1015	-1.0946
	0.4375	0.5125	0.5087	0.4993	0.4349	0.3323	0.3238
	-0.3594	-0.43171	-0.4285	-0.4206	-0.3614	-0.2780	-0.2728
	0.1719	0.2255	0.2237	0.2197	0.1888	0.0913	0.0890
	0.3125	0.2728	0.2706	0.2634	0.2284	0.1104	0.1077
	-0.2764	-0.2407	-0.2387	-0.2324	-0.1398	0	0
	0.1360	0.1271	0.1158	0.1070	0	0	0
	-0.0769	-0.1643	-0.0820	-0.0602	0	0	0
	0.0137	0.1009	0.02405	0.00684	0	0	0
		-0.0403	-0.0088	0	0	0	0
		0.0520	0.0114	0	0	0	0
		-0.0222	-0.0049	0	0	0	0
		-0.0193	-0.0042	0	0	0	0
$\hat{\theta}_\lambda$		0.0145	0.0032	0	0	0	0
		-0.0087	-0.0019	0	0	0	0
		0.0034	0	0	0	0	0
		0.0010	0	0	0	0	0
		0.0037	0	0	0	0	0
		-0.0062	0	0	0	0	0
		-0.0001	0	0	0	0	0
	1.0000	0.9933	0.9933	0.9933	0.9920	0.9803	0.9779
	0.2500	0.2210	0.2210	0.2210	0.2207	0.2181	0.2176
	-0.2500	-0.2281	-0.2281	-0.2281	-0.2213	-0.0100	0
		0.0030	0.0022	0	0	0	0
		-0.0056	-0.0040	0	0	0	0
		0.0020	0.0014	0	0	0	0
		0.0224	0.0163	0	0	0	0
		0.0189	0.0063	0	0	0	0
		-0.0680	-0.0041	0	0	0	0
		-0.0305	0	0	0	0	0
fit_λ		97.36	98.27	98.71	95.24	84.97	84.75

long as $n_a + n_b$ is constant. The automatic choice given by `selstruc` yields the correct model order (18).

There is a slight performance difference, in terms of comparing complexity, for the low model orders used here in favor for the SITB implementation. But as the number of parameters grow, the favor turns to the NNG method. The reason for this is that the complexity of an exhaustive search, as implemented in SITB, is quadratic in the number of models, while the complexity of the NNG method is linear in the number of parameters.

C. Fan and plate data

In this final example, real life data collected from a "Fan and Plate" process, shown in Figure 3, is used. This simple process is used as a laboratory exercise in the undergraduate modeling course at Linköping University. The process input is the motor's driving voltage and the output is a gauge voltage which is proportional to the angle of the plate.

For the identification, the input is chosen as random binary noise of length 1000 with sampling time of 0.04 s.

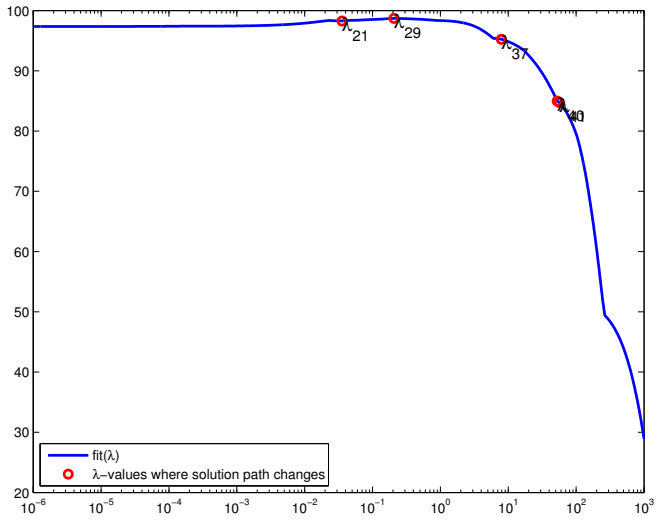

Fig. 2. The fit$_\lambda$ values for the breakpoints and 1000 uniformly logarithmic distributed values of λ, calculated on validation data, for the simulation in Section IV-B.

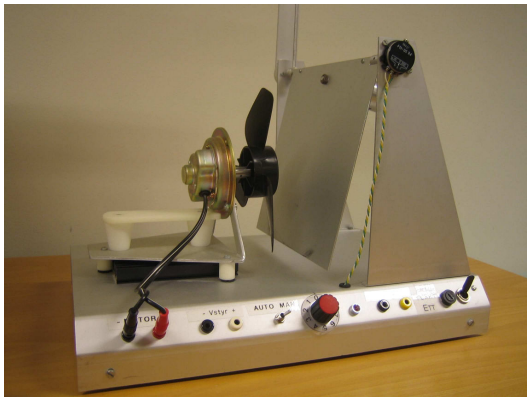

Fig. 3. The fan and plate process.

Two thirds of the samples are used for estimation and the remaining samples are used for validation.

The input delay was determined to be $n_k = 2$. Estimating an initial ARX(50,50,2) model and plugging it into the modified NNG algorithm, results in the one step ahead prediction fit values (19), on validation data, are shown in Figure 4. The choices of the AIC and MDL selection criteria , see e.g. [4], which is an ARX(9,3,2) model, are also presented.

Using the SITB toolbox, letting n_a and n_b roam free up to order 50, via the `arxstruc` command, the MDL and the AIC criterion choices is an ARX(7,4,2) model.

For validation, the simulation results on validation data for the different models is presented in Figure 5. The NNG algorithm has only been used for order selection, i.e. the models have been re-estimated using the `arx` command for the corresponding model orders.

The MDL criterion evaluated on the NNG results yields a recommended model with comparable order and a similar performance compared to the choice when MDL was used together with the `arxstruc` command.

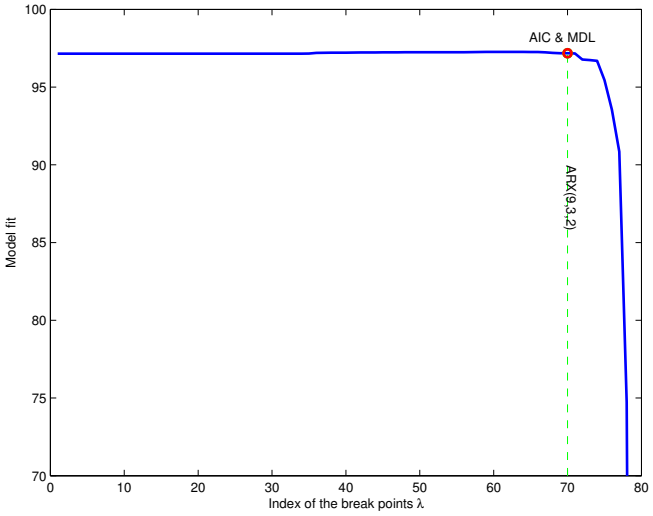

Fig. 4. The one step ahead prediction model fit value (19) for the modified NNG algorithm on fan and plate validation data. The horizontal axis is the index i for the break points λ_i.

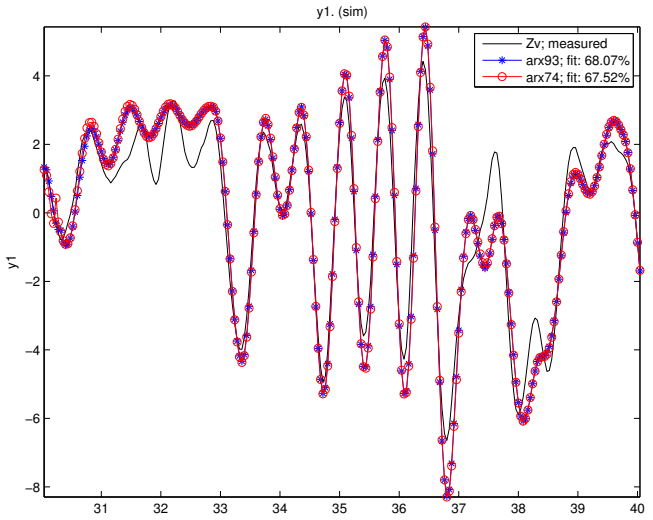

Fig. 5. Simulation for the different model orders on validation data. The NNG choices has been re-estimated on the estimation data.

V. CONCLUSIONS AND FUTURE WORK

In this paper, a new method for order selection of ARX models was presented and an efficient algorithm given. The method is a modified variant of the NNG method [3], where constraints on the weights were added according to the natural order of the regressors in ARX models. Different examples were given, both on data from simulations and from a real life process, and the results look promising, both for model reduction and finding the true model order.

The regularization parameter λ was used to penalize both the pole and the zero polynomials, thus letting the ordering of the output lags and the input lags be independent. This could be extended by splitting the $J(w)$ in (10) into two sums with two different λ, one for weights on the poles and one for the zeros. This would

need to make use of a multiparametric programming algorithm and although they are not as simple as the one presented for a single parameter, efficient algorithms exist [8].

The new method is easily extended to the multivariable case. Here, one could still use one regularization parameter λ and it would be interesting to observe the behavior of the algorithm.

References

[1] K. Åström and P. Eykhoff. System identification – a survey. *Automatica*, 7:123–162, 1971.

[2] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[3] L. Breiman. Better subset regression using the nonnegative garrote. *Technometrics*, 37(4):373–384, 1995.

[4] L. Ljung. *System Identification, Theory for the User*. Prentice Hall, 1999.

[5] S. Niu, L. Ljung, and Å. Björck. Decomposition methods for solving least-squares parameter estimation. *IEEE Transactions On Signal Processing*, 44(11):2847–2852, 1996.

[6] J. Roll. Piecewise linear solution paths for parametric piecewise quadratic programs with application to direct weight optimization. Technical Report LiTH-ISY-R-2816, Linköpings Universitet, 2007. To appear in Automatica.

[7] R. Tibshirani. Regression shrinkage and selection via the lasso. *J. R. Statist. Soc. B*, 58(1):267–288, 1996.

[8] P. Tøndel, T.A. Johansen, and A. Bemporad. Further results on multiparametric quadratic programming. In *Conference on Decision and Control*, pages 3173–3178, Maui, Hawaii, 2003.

[9] M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *J. R. Statist. Soc. B*, 68(1):49–67, 2006.

[10] K. Zhoe, J. C. Doyle, and K. Glover. *Robust and Optimal Control*. Prentice Hall, 1995.