Results on the PASCAL challenge "Simple causal effects in time series"*

Ivan Markovsky

School of Electronics and Computer Science
University of Southampton, SO17 1BJ, UK
Email: im@ecs.soton.ac.uk

Abstract

A solution to the PASCAL challenge "Simple causal effects in time series" (www.causality.inf.ethz.ch) is presented. The data is modeled as a sum of a constant-plus-sin term and a term that is a linear function of a small number of inputs. The problem of identifying such a model from the data is nonconvex in the frequency and phase parameters of the sin and is combinatorial in the number of inputs. The proposed method is suboptimal and exploits several heuristics. First, the problem is split into two phases: 1) identification of the autonomous part and 2) identification of the input dependent part. Second, local optimization method is used to solve the problem in the first phase. Third, ℓ_1 regularization is used in order to find a sparse solution in the second phase.

Keywords: system identification, sparse approximation, ℓ_1 regularization.

1 The proposed model and the corresponding identification problem

The given data in the PASCAL challenge "Simple causal effects in time series" [4] is in the form of two vector time series

$$u_{\mathrm{d}}(1),\ldots,u_{\mathrm{d}}(T)\in\mathbb{R}^m \qquad \text{and} \qquad y_{\mathrm{d}}(1),\ldots,y_{\mathrm{d}}(T)\in\mathbb{R}^p,$$

where u_{d} is referred to as an input (cause) and y_{d} is referred to as an output (effect). The subscript "d" denotes "data" and is used in this paper to distinguish the *given* time series $(u_{\mathrm{d}},y_{\mathrm{d}})$ from a *general* trajectory (u,y) of a model. In $(u_{\mathrm{d}},y_{\mathrm{d}})$, the number of time samples is $T=1095$, the number of inputs is $m=1000$, and the number of outputs is $p=100$. The inputs are binary, i.e., $u_{\mathrm{d}}(t)\in\{0,1\}^m$, however, we do not exploit this feature of the data and the proposed method is applicable for real valued input data.

According to the problem specification, each output y_j has a corresponding *baseline component*, denoted by $y_{\mathrm{bl},j}$, that is periodic and slowly changing and a second component that is determined by a small number (between 0 and 50) of inputs. We model the baseline as an offset-plus-sin in order to capture the mean value and (slow) periodicity in the data. The baseline component is given by the following autonomous model

$$y_{\mathrm{bl},j}(t)=b_j+c_j\sin(\omega_j t+\phi_j), \tag{aut}$$

where $(b_j,c_j,\omega_j,\phi_j)$ are parameters of the model. In order to emphasize the fact that $y_{\mathrm{bl},j}(t)$ depends on the parameters b_j, c_j, ω_j, and ϕ_j, occasionally we write $y_{\mathrm{bl},j}(t;b_j,c_j,\omega_j,\phi_j)$.

We model the component that is determined by the inputs as a linear function $y_j(t)=Au(t)$. The matrix A corresponds to what is called influence matrix in the problem description (see www.causality.inf.ethz.ch). With this choice, the overall model is an affine function of the inputs

$$y_j(t)=y_{\mathrm{bl},j}(t;b_j,c_j,\omega_j,\phi_j)+Au(t). \tag{m}$$

The prior knowledge that only a few inputs affect each output translates to the constraint that the parameter $A\in\mathbb{R}^{p\times m}$ is a sparse matrix, i.e., a matrix with many zero elements.

Note 1. In system theoretic terms, the input-output relation (m) is a static model that decouples into p independent single output multi inputs submodels. The overall model, however, is not static because the baseline is a response of an autonomous dynamical system (of order 3). It is linear in the parameters b_j, c_j, A, and nonlinear in ω_j and ϕ_j.

*Techincal report 16779, ECS, University of Southampton. http://eprints.ecs.soton.ac.uk/

Define

$$Y := \begin{bmatrix} y(1) & \cdots & y(T) \end{bmatrix}, \qquad Y_{\mathrm{bl}} := \begin{bmatrix} y_{\mathrm{bl}}(1) & \cdots & y_{\mathrm{bl}}(T) \end{bmatrix}, \qquad \text{and} \qquad U := \begin{bmatrix} u(1) & \cdots & u(T) \end{bmatrix}.$$

(":=" stands for "left hand side is defined to be equal to the right hand side.") With this notation, the model (m) is written more compactly as the following matrix equation

$$Y = Y_{\mathrm{bl}}(b, c, \omega, \phi) + AU. \tag{M}$$

Here again we explicitly show the dependence of Y_{bl} on the

- offsets $b := (b_1, \ldots, b_p)$,
- amplitudes $c := (c_1, \ldots, c_p)$,
- frequencies $\omega := (\omega_1, \ldots, \omega_p)$, and
- phases $\phi := (\phi_1, \ldots, \phi_p)$.

We define the matrices Y_{d} and U_{d}, constructed from the given data similarly to Y and U. An identification problem corresponding to (M) is

$$\begin{array}{ll} \text{minimize} & \text{over } b, c, \omega \in \mathbb{R}^p,\ \phi \in [-\pi, \pi]^p, \text{ and } A \in \mathbb{R}^{p \times m} \quad \|Y_{\mathrm{d}} - Y_{\mathrm{bl}}(b, c, \omega, \phi) - AU_{\mathrm{d}}\| \\ \text{subject to} & A \text{ is a sparse matrix; in particular, each row of } A \text{ has at most 50 nonzero elements.} \end{array} \tag{P}$$

Problem (P) is not completely specified because the norm used to measure the approximation error is not specified and the sparsity constraint is vague. Independent of the choice of norm, however, problem (P) is a nonconvex optimization problem because the parameters ω and ϕ enter nonlinearly in the model (M), and the sparsity constraint (taken to mean "less than 50 nonzero elements in each row of A") is combinatorial. In addition, the data $(u_{\mathrm{d}}, y_{\mathrm{d}})$ in the PASCAL challenge makes problem (P) a medium size optimization problem, so that computing a global solution is not feasible.

In what follows, we describe heuristics for finding a suboptimal solution of problem (P).

- First, we minimize the fitting criterion $\|Y_{\mathrm{d}} - Y_{\mathrm{bl}}(b, c, \omega, \phi) - AU_{\mathrm{d}}\|$ over b, c, ω, and ϕ, assuming that $A = 0$, i.e., we fit the data by a constant-plus-sin model without considering the effect of the inputs.
- Second, we apply the ℓ_1 heuristic for fining sparse solution A to problem (P), with the parameters ω and ϕ of the Y_{bl} term fixed to the value computed in the first step.

2 Identification of the autonomous term

In this section, we solve the problem

$$\begin{array}{ll} \text{minimize} & \text{over } b, c, \omega \in \mathbb{R}^p \text{ and } \phi \in [-\pi, \pi]^p \quad \|Y_{\mathrm{d}} - Y_{\mathrm{bl}}(b, c, \omega, \phi)\|_{\mathrm{F}}. \end{array} \tag{P1}$$

Here we have chosen the fitting criterion to be the Frobenius norm

$$\|E\|_{\mathrm{F}} := \sqrt{\textstyle\sum_{i=1}^{m} \sum_{j=1}^{n} e_{ij}^2}$$

of the fitting error $E := Y - Y_{\mathrm{bl}}$. Note that problem (P1) decouples into p independent problems

$$\begin{array}{ll} \text{minimize} & \text{over } b_j, c_j, \omega_j \in \mathbb{R} \text{ and } \phi_j \in [-\pi, \pi] \quad \|y_{\mathrm{d},j} - y_{\mathrm{bl},j}(b_j, c_j, \omega_j, \phi_j)\|_2, \end{array} \tag{p1}$$

where $y_{\mathrm{d},j}$ is the jth row of Y_{d} and $y_{\mathrm{bl},j}$ is the jth row of Y_{bl}, i.e.,

$$y_{\mathrm{bl},j}(b, c, \omega, \phi) := \begin{bmatrix} b_j + c_j \sin(\omega_j 1 + \phi_j) & b_j + c_j \sin(\omega_j 2 + \phi_j) & \cdots & b_j + c_j \sin(\omega_j T + \phi_j) \end{bmatrix}.$$

A generalization of this problem to sum of n-sinusoids is well studied in signal processing, where, it is known as *harmonic retrieval* and *line spectral estimation*. For an outdated but nicely written overview of the relevant literature, we refer the reader to [3]. For our purposes, it suffices to say that the methods split into suboptimal heuristics (subspace-type methods) and methods based on local optimization (maximum likelihood-type methods).

For the problem at hand, we use a method from the local optimization family because this approach allow us to fix the frequency to a predefined value chosen from empirical observation of the data. For all $j = 1, \ldots, p$, the frequency ω_j seems to be equal to either $12\pi/T$ (one year period) or $6\pi/T$ (half year period). We solve two problems with $\omega_j = 12\pi/T$ and $6\pi/T$ choose the fit that leads to a smaller value of the cost function. Thus we eliminate ω_j from the optimization problem (p1).

The idea described next is called the *variable projections method*. For a given value of ϕ_j, problem (p1) is a least squares problem in b_j and c_j and therefore it can be solved efficiently and reliably. However, the minimization over ϕ_j still remains to be carried out. This is the difficult part of the problem, because it is a nonlinear least squares problem with respect to ϕ_j.

Alternatively, the linear least squares problem (geometrically a projection) gives us the cost function's value of the latter problem with ϕ_j as the only optimization variable.

For the actual implementation, we use general purpose scalar local optimization solver (the `fminbnd` function of MATLAB's Optimization Toolbox) where in addition it is possible to specify lower and upper bounds on the optimization variable $\phi_j \in [-\pi, \pi]$. For initial approximation we choose $\phi_j = 0$. The results of concrete simulation examples are shown in Figure 1. The implementation of (p1) in the software accompanying this paper is `fit_sin`. By running the script `test_fit_sin`, the reader can verify that for the data in the challenge the method works without apparent problems. Such problems can in general occur due to convergence to a local minimum of (p1).

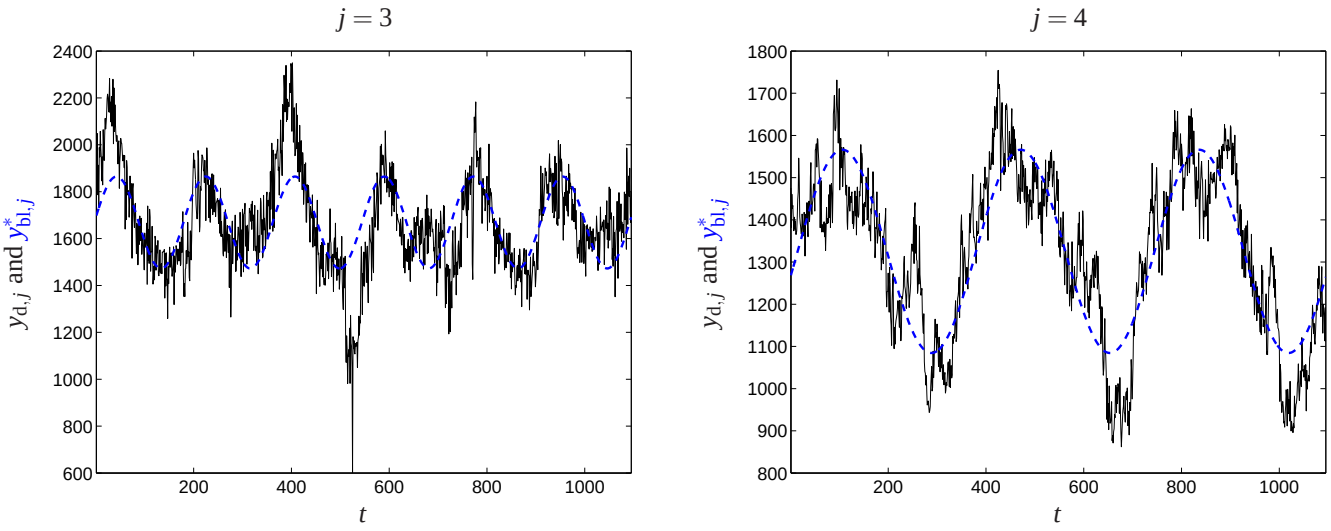

Figure 1: Examples of offset-plus-sin fits produced by `fit_sin`.
Solid line — $y_{\mathrm{d},j}$ (left plot $j = 3$, right plot $j = 4$), dashed line — (locally) optimal fit $y^*_{\mathrm{bl},j}$.

3 Identification of the term involving the inputs

Let ϕ^*, ω^* be a (locally) optimal solution of (P1). The problem we are aiming to solving in this section is

$$\text{minimize} \quad \text{over } b, c \in \mathbb{R}^p \text{ and } A \in \mathbb{R}^{p \times m} \quad \|Y_{\mathrm{d}} - Y_{\mathrm{bl}}(b, c, \phi^*, \omega^*) - AU_{\mathrm{d}}\|_{\mathrm{F}} \quad \text{subject to} \quad A \text{ is sparse.} \tag{P2}$$

Note that the difference between the original problem (P) and problem (P2) is that the ϕ and ω parameters are fixed to the values ϕ^* and ω^* computed in advance (from problem (P1)). Obviously, this step of fixing optimization variables leads to suboptimality, however, it simplifies the problem and is an intuitively justifiable heuristic.

The main difficulty in solving problem (P2) is the sparsity constraint. The approach we adopt is based on the ℓ_1 heuristic for producing sparse solutions, also known in the statistical literature as the *lasso method*. Note that, similarly to the situation in Section 2, problem (P2) is row-wise separable:

$$\text{minimize} \quad \text{over } b_j, c_j \in \mathbb{R}^p \text{ and } a_j \in \mathbb{R}^m \quad \|y_{\mathrm{d},j} - y_{\mathrm{bl},j}(b_j, c_j, \phi_j^*, \omega_j^*) - a_j^\top U_{\mathrm{d}}\|_2 \quad \text{subject to} \quad a_j \text{ is sparse.} \tag{p2}$$

In the rest of this section, we describe two heuristics for finding efficiently a suboptimal solution to problem (p2).

3.1 ℓ_1-regularization of the cost function

We replace the sparsity constraint in problem (p2) by addition of the regularization term $\lambda_j \|a_j\|_1$ in the cost function. The resulting problem is

$$\text{minimize} \quad \text{over } b, c \in \mathbb{R}^p \text{ and } a_j \in \mathbb{R}^m \quad \|y_{\mathrm{d},j} - y_{\mathrm{bl},j}(b, c, \phi^*, \omega^*) - a_j^\top U_{\mathrm{d}}\|_2 + \lambda_j \|a_j\|_1. \tag{p2'}$$

The scalar $\lambda_j > 0$ is a parameter that controls the accuracy vs sparsity trade-off. For large λ_j, the solution a_j computed by problem (p2), tends to have many zero elements. Achieving less than 50 nonzero elements (see the constraint of problem (P)) requires choosing "sufficiently large" value of λ_j. Too large values, however, produce poor fit and are therefore not desirable. The right trade-off between accuracy and sparsity is problem dependent and the choice of λ_j is not automated. The value of λ_j is chosen for the specific data $(u_{\mathrm{d}}, y_{\mathrm{d}})$ by trail and error.

Problem (p2') is a convex optimization problem and therefore can be solved efficiently. We used CVX, a package for specifying and solving convex programs [1, 2], in order to translate (p2') to standard convex optimization problem and solve it by existing software. The computational engines, used by CVX are the SDPT3 and SeDuMi. Since the numerically computed solution has small but nonzero elements, we truncate elements of a_j that are below the convergence tolerance used by the solver to zero.

Figure 2 shows examples of fits obtained with models (m) identified by solving problems (p1) and (p2'). The model corresponding to the left plot involves 4 inputs and the relative fitting error

$$e_j := \frac{\|y_{\mathrm{d},j} - y_{\mathrm{bl},j} - a_j^\top U_\mathrm{d}\|_2}{\|y_{\mathrm{d},j}\|_2} \qquad (*)$$

(i.e., a normalization of the cost function) is $e_3 = 0.0622$. The model corresponding to the right plot involves 2 inputs and the fitting error is $e_4 = 0.0624$. Comparing the results in Figure 1 with those in Figure 2, it is visible how some features of the data that are not captured by the autonomous part of the model are fitted by the part involving the inputs.

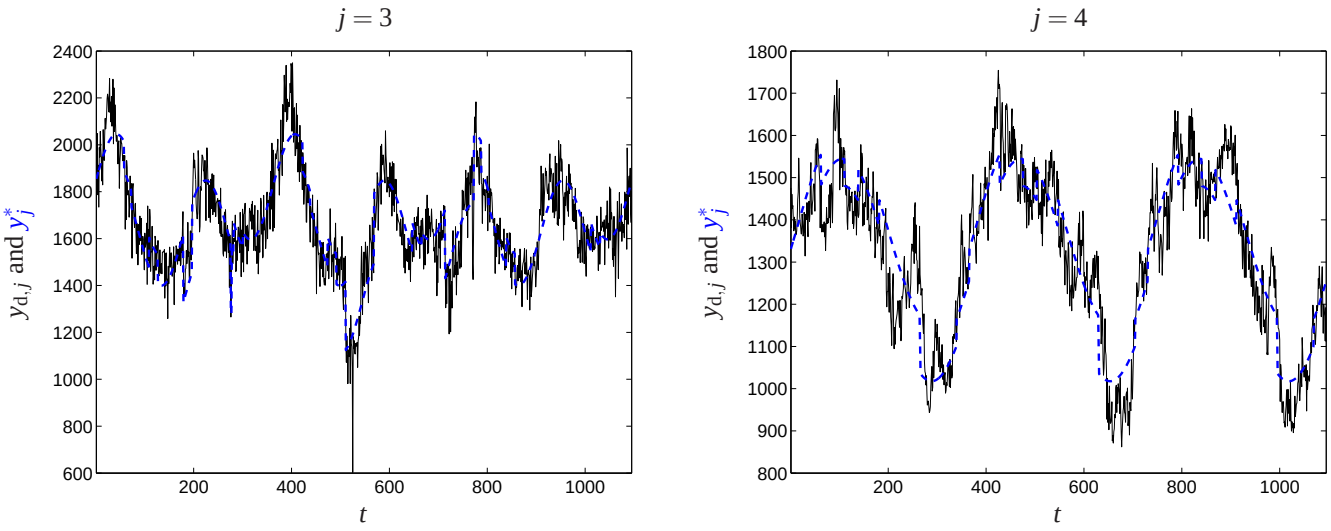

Figure 2: Examples of fits obtained by the overall model (m), identified solving problems (p1) and (p2').
Solid line — $y_{\mathrm{d},j}$ (left plot $j = 3$, right plot $j = 4$), dashed line — suboptimal fit y_j^*.

3.2 ℓ_1-regularization of the cost function

The main difficulty in using (p2') as a heuristic for (p2) is the user choice of the regularization parameter λ_j. In cases when one solves a single instance of the problem, one can afford to plot the sparsity vs accuracy trade-off curve and choose visually from the curve a suitable value for λ_j. In the PASCAL challenge, however, the goal is to solve 100 instances of problem (p2), so that the manual selection of λ_j becomes time consuming (and boring). In addition, it involves the human as a part of the solution, which from our point of view is not acceptable: we are looking instead for an automated solution, which does not involve user defined parameters, that are determined by trail and error.

Motivated by the difficulty of choosing the λ_j parameter (p2'), we propose the following alternative heuristic for finding a suboptimal solution of (p2)

$$\text{minimize} \quad \text{over } b_j, c_j \in \mathbb{R}^p \text{ and } a_j \in \mathbb{R}^m \quad \|y_{\mathrm{d},j} - y_{\mathrm{bl},j}(b_j, c_j, \phi_j^*, \omega_j^*) - a_j^\top U_\mathrm{d}\|_2 \quad \text{subject to} \quad \|a_j\|_1 \leq \gamma_j. \qquad \text{(p2")}$$

In (p2"), the ℓ_1 regularization is given as a bound on $\|a_j\|_1$ rather than as a term in the cost function. The scalar $\gamma_j > 0$ plays a similar role to the one λ_j plays in (p2'). Moreover, there is a one-to-one relation between λ_j and γ_j (duality theory).

Contrary to λ_j, however, $\gamma_j > 0$ has a direct interpretation as a bound on the solution size. This allows us to propose a simple and computationally inexpensive way of choosing γ_j automatically. Suppose that we are aiming to compute a sparse solution with at most k nonzero elements. We choose randomly k rows of U_d (e.g., in the reported simulation results, we choose the first k rows) and let U_d' be the matrix of the chosen rows stacked under each other. Then we take

$$\gamma_j := \|(y_{\mathrm{d},j} - y_{\mathrm{bl},j})U_\mathrm{d}'^{+}\|_1 \qquad (**)$$

where $U_\mathrm{d}'^{+}$ is the pseudo inverse of U_d'. Obviously $(**)$ is heuristic because of the random selection that it involves, however, it has the justification that $(y_{\mathrm{d},j} - y_{\mathrm{bl},j})U_\mathrm{d}'^{+}$ is a particular solution with k nonzero elements.

Figure 3 shows examples of fits obtained with models (m), identified by solving problems (p1) and (p2”). The model corresponding to the left plot involves 24 inputs and the fitting error is $e_3 = 0.0518$. The model corresponding to the right plot involves 25 inputs and the fitting error is $e_4 = 0.0387$. In comparison with the results obtained by using (p2’), now the models have more inputs and achieve smaller fitting errors. This is due to the choice of different points on the corresponding trade-off curves.

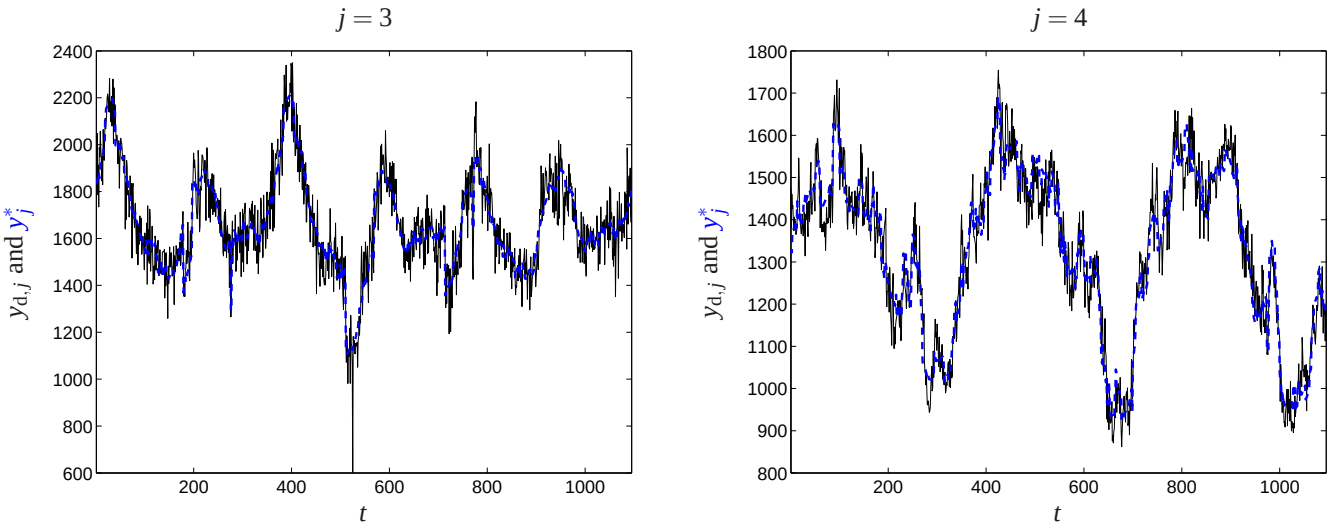

Figure 3: Examples of fits obtained by the overall model (m), identified solving problems (p1) and (p2”).
Solid line — $y_{\mathrm{d},j}$ (left plot $j = 3$, right plot $j = 4$), dashed line — suboptimal fit y_j^*.

4 Nonuniqueness of the solution

For a given baseline Y_{bl}, model (M) is an overdetermined linear system of equations $Y - Y_{\mathrm{bl}} = AU$ in A. Therefore, in order to be able to determine A uniquely from U and $Y - Y_{\mathrm{bl}}$, we need U to be full rank. This condition is not satisfied for the given inputs U_{d} (because $\mathrm{rank}(U_{\mathrm{d}}) = 862$). Taking into account the sparsity constraint, however, we may still be able to recover a unique solution. This would be the case if any nontrivial element in the left null space of A is a dense vector. (Indeed, if a dense vector in the left null space is added to a sparse particular solution, the resulting solution would be dense.)

We consider two special cases that lead to rank deficiency of U:

- *Zero inputs* can not affect the output and therefore removing them from the model leads to an equivalent reduced model $Y - Y_{\mathrm{bl}} = A_{\mathrm{red}} U_{\mathrm{red}}$. Going back from the reduced model to the original model can be done by inserting arbitrary columns in A_{red} at the places of the removed inputs. In order to get as sparse solution as possible, however, we have to insert zero columns in A_{red}. The interpretation of this augmentation is to declare that *according to the model* certain inputs (the zero inputs $u_{\mathrm{d},j}$) to have no effect on the outputs.

- *Inputs that are multiples of other inputs*[1] lead to essential nonuniqueness of the solution, which can not be recovered by the sparsity constraint. Let $u_k = u_l$ for some k and l. Then the model

$$Y - Y_{\mathrm{bl}} = \begin{bmatrix} * & \cdots & * & w & * & \cdots & * & v & * & \cdots & * \end{bmatrix} U,$$

  where w is in the kth column and v is the lth column of A and the model

$$Y - Y_{\mathrm{bl}} = \begin{bmatrix} * & \cdots & * & v & * & \cdots & * & w & * & \cdots & * \end{bmatrix} U,$$

  where v is in the kth column and w is the lth column of A are equivalent (have the same sparsity and achieve the same input-output relation). Therefore, there is no reason to prefer one of these models over the other.

Based on the above consideration, we remove from the data in a preprocessing step, zero inputs and multiple copies of the same input, identify (solving problems (p1) and (p2”)) a model with the reduced set of inputs, and in a postprocessing step recover from the reduced model an equivalent model with the original number of inputs. Apart from reducing the size of the problem the pre and postprocessing steps have the advantage of revealing nonuniqueness of the solution.

[1] For binary inputs this means inputs that are exact copies of other inputs.

Note 2. For the data $(u_{\mathrm{d}}, y_{\mathrm{d}})$, it turns out that all inputs that are copies of other inputs are constants 1, i.e., all time active inputs. Such inputs have the net effect of an offset on the output. However, offset is already included in the autonomous part of the model, see (aut), so that we remove all inputs that are constant 1. The convention of including the offset in the autonomous part of the model has the advantage of resolving the nonuniqueness of the solution problem.

5 Results

The authors of the challenge provide an influence matrix W that shows which inputs affect which outputs. In the evaluation of the model obtained by the proposed identification method, summarized in Algorithm 1, we assume that the nonzero elements of W indicate the "true" inputs affecting each output. Table 1 reports the number of inputs affecting each output, according to our method, the true number of inputs affecting each output, and the number of correctly identified inputs by our method. Software reproducing the results is available from: `http://www.ecs.soton.ac.uk/~im/challenge.tar` (The total execution time is about 3 hours on MATLAB 7.3, run on a PC with 2.13GHz CPU.)

Algorithm 1 Algorithm for identification of model (M).

Input: $U_\mathrm{d} \in \mathbb{R}^{m\times T}$ and $Y_\mathrm{d} \in \mathbb{R}^{p\times T}$.

1: *Preprocessing:* detect and remove redundant inputs, i.e., zero rows of U_d and rows that are multiple copies of other rows of U_d. The subsequent steps are applied on the reduced matrix U_d.
2: **for** $j = 1$ to p **do**
3: Let f' be a (local) minimum of (p1) with $\omega_j = 6\pi/T$ and let ϕ_j' be a corresponding optimal solution.
4: Let f'' be a (local) minimum of (p1) with $\omega_j = 12\pi/T$ and let ϕ_j'' be a corresponding optimal solution.
5: **if** $f' < f''$ **then**
Let $\omega_j^* := 6\pi/T$ and $\phi_j^* := \phi_j'$.
6: **else**
Let $\omega_j^* := 12\pi/T$ and $\phi_j^* := \phi_j''$.
7: **end if**
8: Let $\gamma_j := \|(y_{\mathrm{d},j} - y_{\mathrm{bl},j})U_\mathrm{d}(1{:}10,:)^+\|_1$ (here we use MATLAB notation for indexing).
9: Let a_j' be a solution to (p2") with parameters $\phi_j = \phi_j^*$ and $\omega_j = \omega_j^*$.
10: Determine the sparsity pattern of a_j' by truncating elements smaller by the convergence tolerance to zero.
11: Let (b_j^*, c_j^*, a_j^*) be a solution to problem (p2) with the sparsity pattern determined on the previous step and with parameters $\phi_j = \phi_j^*$ and $\omega_j = \omega_j^*$. (This is a linear least squares problem)
12: **end for**
13: *Postprocessing:* add zero rows in A^* that correspond to the removed inputs in the preprocessing step.

Output: $Y_\mathrm{bl}(b^*, c^*, \omega^*, \phi^*)$ and A^*.

output	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
# computed u's	11	1	24	25	33	17	7	32	10	21	34	15	18	21	20	18	15	31
# true u's	12	2	7	39	13	20	23	41	14	23	23	33	30	15	30	46	8	1
# correct u's	2	1	2	7	4	4	5	6	6	7	11	7	9	0	5	5	3	1
output	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
# computed u's	17	22	10	2	16	15	22	36	25	21	7	24	25	11	24	14	8	18
# true u's	17	45	13	2	25	10	24	39	41	30	20	47	23	2	45	26	4	40
# correct u's	6	5	6	2	10	2	6	5	9	8	4	2	7	2	6	3	4	3
output	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54
# computed u's	2	14	58	39	9	10	7	19	23	24	14	18	18	24	15	3	16	15
# true u's	2	33	11	14	31	2	5	48	39	38	13	33	50	24	15	4	22	7
# correct u's	1	5	6	4	5	1	4	8	4	6	5	7	5	5	5	3	5	3
output	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72
# computed u's	31	137	16	5	14	14	1	8	11	3	14	5	15	26	10	3	14	32
# true u's	48	41	47	4	23	34	1	10	34	8	20	12	22	26	16	13	10	50
# correct u's	10	19	8	2	7	4	1	3	6	2	6	5	3	10	2	3	3	9
output	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90
# computed u's	28	4	17	18	23	16	10	13	14	16	7	13	30	12	4	2	14	17
# true u's	42	4	40	36	30	20	10	12	17	15	6	24	27	33	4	5	9	43
# correct u's	10	3	3	8	7	4	5	5	3	4	4	6	5	5	3	2	6	5

output	91	92	93	94	95	96	97	98	99	100	total for the 100 outputs
# computed u's	14	17	14	31	8	15	13	27	28	14	1796
# true u's	36	14	17	41	20	41	34	44	38	16	2321
# correct u's	3	6	5	7	6	7	3	10	10	2	507

Table 1: Results for the model identified by Algorithm 1.
First row — number of inputs affecting an output according the the identified model.
Second row — number of inputs affecting an output according to the influence matrix, supplied by the organizers.
Third row — the number of correctly identified inputs by Algorithm 1.

References

[1] M. Grant and S. Boyd. CVX: Matlab software for disciplined convex programming. `http://stanford.edu/~boyd/cvx`, 2008.

[2] M. Grant and S. Boyd. Graph implementations for nonsmooth convex programs. In V. Blondel, S. Boyd, and H. Kimura, editors, *Recent Advances in Learning and Control*, pages 95–110. Springer, `http://stanford.edu/~boyd/graph_dcp.html`, 2008.

[3] S. Kay and S. Marple. Spectrum analysis–a modern perspective. *Proceedings of the IEEE*, 69(11):1380–1419, 1981.

[4] Causality workbench team. PROMO: Simple causal effects in time series, 08 2008.