Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.Doi Number

Dictionary Morphing Orthogonal Least Squares Regression for NARMAX-based Black-box Fitting

Stéphane Thunus[1], Julian Parker[2], and Stefan Weinzierl[1]

[1]Technische Universität Berlin - Fakultät I, Berlin, 10587, Germany

[2]Native Instruments, Berlin, 10997, Germany

Corresponding author: Stéphane Thunus (Stephane@Thunus.org).

ABSTRACT This paper proposes an optimization pipeline allowing the recursive forward orthogonal least squares regression (rFOrLSR) to morph its regressors to better fit the desired system. The rFOrLSR is a supervised machine learning algorithm generating a sparse symbolic representation of non-linear recursive systems by choosing the optimal regressors from a given set (called dictionary) and their coefficients. A symbolic representation is an equation containing arbitrary non-linear terms, which describes or approximates a system. This paper focusses on rFOrLSR dictionaries containing (permissibly discontinuous) elementwise non-linearities applied on arbitrary regressors. The morphing operation adds scaling coefficients and arbitrary linear combinations of dictionary terms inside the chosen regressor's outer elementwise non-linearity. First, a criterion determining regressor morphability is introduced. Then, a procedure determining what multiplicative coefficients and what supplementary terms to add is presented. This is followed by a local coefficient infinitesimal optimization procedure searching the best coefficients inside the chosen non-linearity. Once all regressors are chosen and potentially morphed, all coefficients are infinitesimally optimized to adjust the entire expression. Thus, the proposed pipeline performs transformations on the user-defined regressor-set to increase fitting precision and model sparsity of NARMAX systems.

INDEX TERMS Forward Orthogonal Least Squares Regression; Model Structure Detection; NARMAX; Non-linear System Identification; Supervised Machine Learning For Parameter Estimation; Symbolic Regression.

I. INTRODUCTION

NARMAX stochastic processes (Non-linear Auto-Regressive Moving Average with eXogenous inputs) have been used to model spatial and temporal non-linear systems in the fields of medicine, chemistry, biology, geography, industrial process control, economics and more [1]. The NARMAX models' application fall into the three categories of model identification, prediction, and approximation.

Model identification emulates a given system to recreate its output for analysis or compensation purposes. To illustrate, having an algebraic representation of a non-linear distortion might allow to trace it back to the component generating it for correction or prevention. NARMAX models are of interest, yielding interpretable non-linear equations consisting of arbitrary terms, as opposed to for example artificial neural networks constituted of large coefficient matrices and non-linearities.

System behavior prediction allows, amongst other things, to prevent the system from harming other (for example electric) components or compress its output for transmission or storage. To illustrate, if a NARMAX model predicts 60% of the temporal or spatial signal's variance from past or surrounding values, only the remaining 40% must be transferred or stored.

Lastly, NARMAX systems can also replace known but computationally intensive processes, as is commonly done with Taylor or Padé expansions.

This paper introduces a new algorithm to a black-box fitting symbolic regression algorithm class named "Forward Orthogonal Least Squares Regression" (FOrLSR /FOLSR/

OFR) [1]. FOrLSR algorithms are dictionary-based symbolic least-squares regressions, as they generate (or are provided with) a "dictionary" of regressors to find the shortest possible regressor sequence describing the system with the desired precision. To illustrate, common expansions like Taylor, Padé, Fourier, RBF and Wavelet are equivalent to fitting the entirety (being non-sparse expansions) of a dictionary containing a single function type (polynomials, oscillations, hyper-ellipsoids, etc). The FOrLSR's dictionary, however, can contain any number of arbitrary spatial and temporal functions and combinations thereof, of which the FOrLSR selects only the most relevant to describe the system.

A further advantage is the least squares framework ensuring optimal fitting for arbitrarily large subspaces, whereas Taylor- and Padé-like expansions are only optimal at their expansion point and can quickly drift off.

The remaining paper is organized as follows: Section II presents the rFOrLSR, while the new arborescent structure and dictionary morphing are respectively introduced in Sections III and IV. Section V provides examples and benchmarks, while Section VI concludes the paper.

II. THE RECURSIVE FORWARD ORTHOGONAL LEAST SQUARES REGRESSION (rFOrLSR)

From a linear algebra perspective, expansions are projections into function spaces with basis vector sequences (BVS) depending on the used expansion (monomials for Taylor, oscillations for Fourier, etc), whose potentially infinite dimension depends on the expansion order. The FOrLSR searches the BVS (model structure detection) representing the given system response vector $\underline{y}$ with the smallest mean squared error and the shortest BVS (model sparsity) [1]. This optimal BVS being system-dependent, thus unknown, a large dictionary D_C containing n_C candidate basis vectors (= regressors) is iteratively scanned to extract the best fitting regressor to add it to the BVS (regression). Thus, the optimal point (regression coefficients $\hat{\underline{\theta}}$) in an optimal subspace (spanned by the BVS) of D_C's columns is searched for. To make this operation computable, one evaluates the regressors, being transformations of the system input, output or noise, on a zero-mean uniform white noise sequence of length p. This corresponds to working in a vector-space spanned by finite dimensional observation vectors rather than a Hilbert space of functions. This reduces the symbolic regression problem to the usual least squares regression, solved with a Gram-Schmidt-based QR-like decomposition augmented with a similarity metric iteratively selecting the most similar basis vector to the desired system's output $\underline{y} \in \mathbb{R}^p$ (see [1]).

Thus, the FOrLSR acts as a linear equation system solver (LESS) decomposing D_C into an orthogonal and an upper unitriangular matrix, both of lower dimensions, yielding a solution vector $\hat{\underline{\theta}}$ containing the original system solution's non-zero entries.

The Gram-Schmidt-based orthogonalization allows a forward regression not required to estimate model parameters at each iteration when adding a regressor to the BVS [1][2]. Thus, the FOrLSR re-orthogonalizes the regressor observations dictionary $D_\mathrm{C} \coloneqq \left[\underline{\varphi}_j\right]_{j=1}^{n_\mathrm{C}} \in \mathbb{R}^{p \times n_\mathrm{C}}$ with respect to all vectors (regressors) $\underline{\psi}_j$ previously added to the BVS (modindeel). The orthogonalization, being the algorithm's most expensive operation, p and n_C being usually very large, it is of interest to vectorize it and make it recursive.

A. THE ORTHONORMAL ANNIHILATOR MATRIX

[Edited] (1)

[Edited] (explanations)

The index set of unused regressors U is an input, as the morphing might require access to $\underline{\varphi}_m$ being function arguments of other regressors, which one might not want to have as selectable regression terms.

ALGORITHM 1: The rFOrLSR with morphing algorithm

[Edited] (pseudo-code)

[Edited] (explanations)

The Section IV's morphing takes place after respectively 2.e and 3.e and replaces $\underline{\omega}_\ell$ and its norm by that of the custom-made regressor. An unorthogonalized version is appended to D_C and, instead of $U[\ell]$, one appends $\mathrm{ncols}(D_\mathrm{C})$ to $\mathcal{L}$, being the new regressor's column number.

[Edited]

III. THE DICTIONARY MORPHING

It is unlikely that dictionaries contain the optimal non-linearities of the form $f_\circledcirc\left(\sum_{j=1}^{r} \xi_j \underline{\chi}_j\right)$, as this would require an quasi-infinite number of infinitesimal increments for each ξ_j and a combinatorial order of $\underline{\chi}_j$ combinations for each $r > 0$ (see Section V-C). Morphing the non-linearities via their arguments to adapt them to the fitted system keeps the dictionary size n_C finite, while increasing model sparsity in addition to fitting precision and speed.

From a linear algebra perspective, the dictionary morphing (r)FOrLSR iteratively spans a vector-space by selecting at each iteration the basis vector $\underline{\varphi}_m$ the most similar to the system response $\underline{y}$, then modifying it to resemble $\underline{y}$ more. This minimizes the amount of information lost due to fitting error by generating an optimal vector-space. Thus, the measurement is adapted to the system, rather than the system being compared to pre-determined aspects. The system is then represented as a single point (regression coefficients $\hat{\underline{\theta}}$) in the vector-space spanned by the selected regressors.

For comparison, classic expansions like Fourier and Laplace fit the entire given dictionary (= imposed terms and expansion order), whereas the (r)FOrLSR selects only the most relevant terms (= system dependent term selection and

expansion order). The next logical step is adapting the selected regressors to the system, making the expansion fully system dependent.

The morphing procedure comprises four steps. After the (r)FOrLSR regressor selection, the regressor is parsed to deduce the non-linearity f and its argument $\underline{\chi}_0$. If the function is morphable, the procedure continues with a genetic vector-space generating algorithm determining how many and which supplementary arguments $\underline{\chi}_j$ improve the fitting. Finally, an infinitesimal optimizer determines the optimal coefficients $\underline{\xi}$ and returns the vector to the rFOrLSR.

The morphing order $\mathfrak{O}$ determines the number of added arguments. Thus, $f_{\circledcirc}(\underline{\chi}_0) \to f_{\circledcirc}(\xi_0\underline{\chi}_0)$ is 0th order morphing and $f_{\circledcirc}(\underline{\chi}_0) \to f_{\circledcirc}(\sum_{j=0}^{n}\xi_j\underline{\chi}_j) = f_{\circledcirc}(X_\ell\underline{\xi})$ nth order. The morphing keeps the FOrLSR philosophy that the function represented by each $\underline{\chi}_j$ is irrelevant, the $\underline{\chi}_j \in \mathbb{R}^p$ being treated as constant vectors. This greatly simplifies the procedure and allows morphing with arbitrarily complex basis vectors. In particular, the functions represented by the $\underline{\chi}_j$ can be discontinuous, non-differentiable, case-dependent and range over multiple time steps.

The FOrLSR's term-selection is described in Section II, while the term parsing depends on the dictionary constructor, being thus implementation dependent. The remaining procedures are described in the following sub-sections.

The used squared correlation is replaceable with any application-dependent similarity metric, whose gradient's and hessian's analytic form is obtainable via Laure *et al.*'s online tensor calculus tool [8].

A. MORPHABILITY

Not all functions are worth morphing. Firstly, adding arguments to the identity is equivalent to adding them directly to the BVS. Secondly, some functions exhibit a property here labelled "relaxed homogeneity", where their added argument coefficient ξ_0 gets absorbed by their regression coefficient $\hat{\theta}$:

Relaxed homogeneity (RH):

[Edited]

To illustrate, powers, fractions and absolute values exhibit the RH property: **[Edited]**

ALGORITHM 2: **A simple algorithm determining morphability**

[Edited]

[Edited] (explanations)

B. THE GENETIC VECTOR-SPACE GENERATOR (GenVSGen)

Once the selected term is parsed and known to be morphable, the optimal arguments $\{\underline{\chi}_j\}_{j=0}^{r-1}$ and their number $r \in \{i\}_{i=1}^{\mathfrak{O}+1}$ can be searched for. The below algorithm is a quasi-grid search illustrating the minimal procedure more advanced genetic algorithms must follow.

[Edited]

ALGORITHM 3: **A basic genetic vector-space generating algorithm**

[Edited] (algorithm)

[Edited] (explanations)

If the optimal $\underline{\xi}$ lies in $[s_0; e_0] \times [-s_1; s_1]^{r-1}$, the correct regressors $X_\ell \coloneqq \left[\underline{\chi}_j \oplus m_j\right]_{j\in(i_0 \Vdash t_{max})}$ are generally found, with, however, no guarantee that the selected $\underline{\xi}_{max}$ is close to the optimal $\underline{\xi}$, which is remedied by the following step.

C. THE SELF-ORTHOGONALIZING INFINITESIMAL OPTIMIZER

The vector-space spanned by GenVSGen's X_ℓ allows infinitesimal optimization, which must construct vectors exhibiting the dictionary's centering and orthogonalization properties. The orthonormal annihilator matrix P_{A} (equation (1)), being equivalent to a Gram-Schmidt orthogonalization iteration, embeds the current FOrLSR state by containing the regressors' selection order, norms and orthogonalized versions. This allows self-orthogonalizing gradient- and hessian-based optimization in the current BVS's orthogonal complement by embedding all necessary information in the optimization problem. The ground truth $\underline{y}$ is also orthogonalized to eliminate the variance explained by previous regressors: $\underline{y}_o \coloneqq P_{\mathrm{A}}\underline{y}$.

In the general case, first-order line searches perform poorly due to the optimization landscape's roughness and narrow meandering valleys. Indeed, the hessian is mostly extremely poorly conditioned (up to 10^8) and should be assumed indefinite for arbitrary non-linearities, which also excludes most second-order line search methods.

A trust region method with Moré and Sorensen's "Nearly Exact" sub-problem solver [9] is an adequate infinitesimal solver. It explicitly uses the hessian, allows it to be indefinite, and is efficient for low dimensional problems. The optimization problem is thus:

Cost Function: [Edited]

Be further $f'_{\circledcirc}$ and $f''_{\circledcirc}$ elementwise functions applying the selected non-linearity's first and second derivatives.

$\underline{\mathcal{C}}(\underline{\xi})$ and $\underline{y}_o$ being per construction mean-free, the gradient and the Hessian simplify to respectively:

Gradient: [Edited]

Hessian: [Edited]

To counteract the optimization function's severe non-convexity resulting in many local minima, the optimizer should be retriggered multiple times while adding uniform noise to GenVSGen's output $\underline{\xi}$. Then, a validation proce-

dure consisting of the regression's mean squared error (MSE) should select the best result. GenVSGen's $\underline{\xi}$ should be taken as baseline to guarantee, similarly to GenVSGen, that this morphing step ameliorates the fitting precision. The MSE computation requires the same steps as the FOrLSR algorithm's termination (steps 3.i, 3.j and 4.b in Algorithm I, Section II-D) in addition to centering the generated regressor beforehand.

Overfitting can be reduced by rounding the coefficients in $\underline{\xi}$ up to a certain precision (quatization).

The selected morphed vector is then processed by the FOrLSR as if taken directly from the dictionary D_{C}. The AOrLSR requires that the vector remains accessible to (at least) the node's children, while the final validation procedure requires $\underline{\xi}$ and the argument- and function-indices to recreate the morphed regressor on different test sequences.

TODO More Supplementary remarks:
[Edited]

D. THE FINE-TUNING OPTIMIZER

Once the rFOrLSR has selected sufficient terms to meet its termination criterion, the whole symbolic expression representing the system is available. Since the term selection and infinitesimal optimization are done greedily, it is beneficial to perform a final optimization step to optimize all coefficients simultaneously. While the self-orthogonalizing infinitesimal optimizer (SOIO) optimizes only the current regressor's morphing coefficients $\underline{\xi}$, this procedure optimizes the expression coefficients $\hat{\underline{\theta}}$ in addition to each regressor's $\underline{\xi}$. Thus, the gradient is the concatenations of all those partial derivatives.

Gradient: [Edited]

Iteration: [Edited]

IV. EXAMPLES

Section V-C illustrates the dictionary morphing's fitting quality improvement and model sparsification and the quasi-impossibility to store a dictionary large enough to provide similar results.

For Sections TODO, the results slightly differ depending on the white noise input sequence $\underline{x}$, thus the most representative of multiple runs was taken.

A. Case Study 1

The following example illustrates the morphing's necessity for composite regressors due to otherwise extreme storage consumption and the model sparsification and precision increase brought by the morphing.

Let system 3 be:

$$\begin{aligned} y[k] \coloneqq\ & 0.4x^2[k] - 0.5y^2[k-2] \\ & + 0.6\cos\{-0.3x[k-1]y[k-1] \\ & + 0.2\,\mathrm{abs}(x[k-2]x[k-3]) \\ & + 0.9\,y[k-1]\} \end{aligned}$$

System 3 is correctly retrieved by a single FOrLSR (= height 0 arborescence) with 2nd order morphing with System 1's D_{C} (see (4)) using an input signal amplitude of 1.5. This expansion nesting NARMAX models has the form

$$y[k] = \sum_{n=1}^{n_{\mathrm{r}}} \theta_n f_n \left(\sum_{j=0}^{r_n-1} \xi_{n,j} h_{n,j} \left(\prod_{i=1}^{e_{n,j}} \varphi_{n,j,i}\left[k - d_{n,j,i}\right] \right) \right)$$

with $f_n, h_{n,j}$ scalar functions and input/output terms $\varphi_{n,j,i} \in \{x, y\}$ with integer delays $d_{n,j,i} \in \{k\}_{k=1}^{5}$, monomial expansion order $e_{n,j} \in \{k\}_{k=1}^{3}$ and regression coefficients $\theta_n \in \mathbb{R}$. Here, $h_{n,j}\left(\prod_{i=1}^{e_{n,j}} \varphi_{n,j,i}\left[k - d_{n,j,i}\right]\right) \mathrel{\widehat{=}} \chi_j \in D_{\mathrm{C}}$ is constant but the morphing correctly sets all h, φ, d, e by choosing the correct dictionary terms.

The dictionary size n_{C} required for the above regression without morphing is as follows: Assuming a coarse grid with $n_{\mathrm{G}} \coloneqq 21$ steps of 0.1 in $[-1,1]$ for each of the $r = 3$ ξ_j, yields 21^r grid points per regressor combination. The non-linearity contains any of the 363 monomial products as starting term $\underline{\chi}_0$, then any dictionary term can be added.

This yields $363 \sum_{r=1}^{3} 21^r \frac{1441!}{(r-1)!(1441-r+1)!} \approx 3.5 \cdot 10^{12}$ linear combinations for each of the $n_{\mathrm{f}} \coloneqq 3$ non-linearities. Using 64bit floats and $p = 2'000$, this requires about 153'000 TB of storage compared to about 22MB for the above D_{C}. This is excessive considering the limited dictionary, the low precision steps of 0.1 and the very reduced argument range of $[-1,1]$. A grid with steps of 0.05 in $[-1,1]$ or steps of 0.1 in $[-2,2]$ ($n_G = 41$) generates about $77.8 \cdot 10^{12}$ linear combinations, thus $1.1 \cdot 10^{6}$TB.

The linear combinations number's growth is approximately $\mathcal{O}\left(n_{\mathrm{G}}^{\Phi+1} \times n_{\mathrm{C}}^{\Phi} \times n_{\mathrm{f}}\right)$, being unfeasible even for small problems.

GenVSGen requires only a very coarse grid ($n_{\mathrm{G}} = 13$ in $[-1.5,1.5]$, thus steps of 0.25 in this example) to recognize the correct terms, while the infinitesimal optimizer achieves any precision and can span arbitrary ranges, via its trust region method following the direction of ascend, if not stuck in a local maximum.

Fitting the above example took GenVSGen about 18 hours (being uncompiled single-thread python on CPU). This is many orders of magnitude longer than the (r)FOrLSR, hence the need for more refined genetic algorithms. First order morphing, however, takes about 4s, making it trackable for larger arborescences.

For expression retrieval, the (r)FOrLSR must select the non-linearity containing one correct argument, requiring the unmorphed regressor to score higher in the similarity metric than other dictionary terms. Thus, for certain functions, a very coarse grid remains necessary. To illustrate, $\cos(x[k])$ and $\cos(5x[k])$ are dissimilar for most metrics and thus

$\cos(x[k])$ wouldn't be selected by the (r)FOrLSR, despite being morphable into the correct term. Thus, D_C must contains terms more similar to $\cos(5x[k])$.

Similarly, low argument variance can make most of the data stay in f's quasi-linear part such that no or the wrong f is chosen by the (r)FOrLSR. To illustrate, if system 3 had all coefficients inside the cosine set to respectively -0.4, 0.4, 0.4, the (r)FOrLSR would chose $x[k-1]y[k-1]$ instead of $\cos(y[k-1])$ as third term.

Fitting system 3 with a single (r)FOrLSR with $\rho = 0.001$ yields the results in the following Table IV, which compares different morphing orders via their BVS length n_r, mean absolute error (MAE), maximal deviation (MD) and the variability measure median absolute deviation (MAD). The last three metrics are expressed in percentage of the output signal $\underline{y}$'s amplitude. To illustrate, first order morphing yields a model with 6 terms which deviates on average 0.37% from the target $\underline{y}$'s amplitude but not more than 4.58% with a median deviation of 0.232%.

TABLE V: The evolution of BVS length n_r, mean absolute error, maximal deviation and median absolute deviation when fitting System 3 with increasing morphing order from no morphing to 2nd order.

Table V	n_r	MAE %	MD%	MAD%
None	9	0.54%	6.64%	0.395
Order 0	9	0.49%	7.17%	0.355
Order 1	6	0.37%	4.58%	0.232
Order 2	3	exact	exact	exact

Similarly to the above example, in most cases, the morphing reduces BVS length and the error metric.

Being a greedy algorithm, the morphing procedure adds terms to the non-linearity to maximize each morphable term's similarity to the system output $\underline{y}$. In some cases, however, this results in suboptimal BVS, as multiple morphed terms share and fit the variance of what would otherwise be covered by a single dictionary term. This is the same greediness problem discussed at Section III's start and in [2] and is thus similarly mitigated by using deeper arborescences or low morphing orders.

B. Case Study 2

TODO: RBF example

C. Case Study 3

TODO: Wavelet/ sparse Fourier example

V. CONCLUSION

The first proposed improvement is the orthonormal annihilator matrix P_A, which implicitly contains the current regression state.

The proposed orthonormal annihilator matrix P_A further allows computing genetic vector-space generating algorithms and self-orthogonalizing infinitesimal optimizers to morph regressors. This allows to generate custom regressors for the current system, which would require a quasi-infinite dictionary size, if generated in advance.

These modifications achieve sparser, lower error and more complex NARMAX expansions.

The first authors' python AOrLSR / DMOrLSR library is available at https://github.com/Stee-T/rFOrLSR.

Further research in the NARMAX-expansions aspect should go towards more efficient genetic vector space generating algorithms and potentially re-morphing already morphed regressors at each added regressor. Furthermore, closed-form gradient- and hessian-expressions for more advanced similarity metrics and for nested functions might be of interest. Finally, the rich literature on external parameter support such as [11][12][13], would allow modulating the expansion's parameters to model system changes.

AKNOWLEDGMENT

The first author thanks Paolo Combes for his thorough proof-reading and feedback.

REFERENCES

TODO add all other sources

[1] S. A. Billings, "Model Structure Detection And Parameter Estimation", *Nonlinear System Identification: NARMAX Methods in the Time, Frequency, and Spatio-Temporal Domains.* s.l.:John Wiley & Sons, Ltd.. 2013, chapter 3, pp. 64-104.

[2] Y. Guo , L.Z. Guo, S. A. Billings, H.-L. Wei, Ultra-Orthogonal Forward Regression Algorithms for the Identification of Non-Linear Dynamic Systems. *Neurocomputing letters*, 2015 DOI: 10.1016/j.neucom.2015.08.022i.

[3] H.-L. Wei , S. A. Billings, Sparse Model Identification Using a Forward Orthogonal Regression Algorithm Aided by Mutual Information. *IEEE Transactions on Neural Networks*, 18(1), pp. 306 - 310, 2007, DOI: 10.1109/TNN.2006.886356.

[4] H.-L. Wei, S. A. Billings, Model structure selection using an integrated forward orthogonal search algorithm assisted by squared correlation and mutual information. International Journal of Modelling Identification and Control, 3 (4). pp. 341-356. 2008, DOI: 10.1504/IJMIC.2008.020543

[5] S. Laue, M. Mitterreiter, J. Giesen, , A Simple and Efficient Tensor Calculus. New York, *Conference on Artificial Intelligence, (AAAI)*, 2020.

[6] J. J. Moré, D. C. Sorensen, Computing a Trust Region Step. *Siam Journal on Scientific and Statistical Computing*, Band 4, pp. 553-572. 1983, DOI: 10.1137/0904038

[7] Q.M. Zhu, An implicit least squares algorithm for nonlinear rational model parameter estimation, Applied Mathematical Modelling, 29, pp 673-689. 2004. DOI: 10.1016/j.apm.2004.10.008

[8] H.-L. Wei, Y. Gu, A robust model structure selection method for small sample size and multiple datasets problems. Information Sciences, Band 451-452, pp. 195-209, 2018, DOI: 10.1016/j.ins.2018.04.007.

[9] Y. Li, H.-L. Wei, Billings S. A., P.G. Sarrigiannis, Identification of nonlinear time-varying systems using an online sliding-window and common model structure selection (CMSS) approach with applications to EEG, International Journal of Systems Science, 2015, DOI: 10.1080/00207721.2015.1014448

[10] A. Kadochnikova, Y. Zhu, Z. -Q. Lang and V. Kadirkamanathan, "Integrated Identification of the Nonlinear Autoregressive Models With Exogenous Inputs (NARX) for Engineering Systems Design," in IEEE Transactions on Control Systems Technology, 2022, DOI: 10.1109/TCST.2022.3171130.

APPENDIX 1

This appendix describes the notation used throughout the paper. Vectors are denoted with underlined lower-case letters ($\underline{a}$), matrices and sets with uppercase letters (A) and scalars with lower-case letters (a). Set and tensor constructors are $\{a_j\}_{j=1}^{n} \coloneqq \{a_1, \dots, a_n\}$, $[a_j]_{j=1}^{n} \coloneqq [a_1, \dots, a_n]$ (row vectors) and $[\underline{a}_j]_{j=1}^{n}$(matrices). They also support index sets: $[\underline{a}_j]_{j\in S} \triangleq [\underline{a}_{s_1}, \dots, \underline{a}_{s_{|S|}}]$ with $S \triangleq \{s_j\}_{j=1}^{|S|}$ and $|S|$ the set cardinality. Be further $[a_j]_{j=1}^{n,\downarrow} \coloneqq \left([a_j]_{j=1}^{n}\right)^T$ a vertical ($\downarrow$) tensor constructor and the horizontal concatenation operator $\nparallel$. For disambiguation, elementwise tensor functions are denoted with a $\circledcirc$, such as $\underline{a}^{\circledcirc n} \coloneqq [a_j^n]_{j=1}^{\dim(\underline{a}),\downarrow}$ and $f_{\circledcirc}(\underline{a}) \coloneqq [f(a_j)]_{j=1}^{\dim(\underline{a}),\downarrow}$. Elementwise or broadcasted tensor arithmetic operation symbols are in a circle such as $\underline{a} \oslash \underline{b} \coloneqq [a_j/b_j]_{j=1}^{\dim(\underline{a})=\dim(\underline{b}),\downarrow}$, $\underline{a} \oplus b \coloneqq [a_j + b]_{j=1}^{\dim(\underline{a}),\downarrow}$ or $A \odot \underline{b} \coloneqq [\underline{a}_j \circ \underline{b}]_{j=1}^{\mathrm{ncols}(A)}$ with $\underline{a}_j \circ \underline{b}$ a Hadamard product. Tensor slicing is denoted with vector indexation such as $A[[j]_{j=1}^{s}, s] \coloneqq [a_{j,s}]_{j=1}^{s,\downarrow}$. Matrix columns are denoted by lower-casing the matrix name and adding their index such as $\underline{\omega}_j \in \Omega$ and $\underline{d}_{\mathrm{S},j} \in D_{\mathrm{S}}$.

APPENDIX 2

TODO add Gradient; Hessian computation.