Inverse Static Analysis of Massive Parallel Arrays of Three-State Actuators via Artificial Intelligence

Felix Pasila[*§], Rocco Vertechy[†],
Giovanni Berselli[‡] and Vincenzo Parenti Castelli[*]

[*] Dept. of Mech. Eng., University of Bologna, Italy
[§] Dept. of Elec. Eng., Petra Christian University, Indonesia
[†] Percro Laboratory, Scuola Superiore Sant'Anna, Pisa, Italy
[‡] Dept. of Mech. Eng., University of Modena and Reggio Emilia, Italy

Abstract Massive parallel arrays of discrete actuators are force-regulated robots that undergo continuous motions despite being commanded through a large but finite number of states only. Real-time control of such systems requires fast and efficient methods for solving their inverse static analysis, which is a challenging problem. Artificial intelligence methods are investigated here for the on-line computation of the inverse static analysis of a planar parallel array featuring eight three-state force actuators and possessing one degree of revolute motion.

1 Introduction

Discrete-State Manipulators (DSM) are a special kind of mechanisms whose actuators can be made switching among a finite number of states only. Introduced in the early 1970's [1] in an attempt to conceive sensor-less robots as well as to reduce the complexity of control systems and computer interfacing, nowadays DSM can be classified into two different groups depending on whether their actuators act as either discrete displacement generators [2-5] or discrete force generators [6]. This work deals with the latter type of DSM, usually referred to as Massively Parallel Robots (MPR). In essence, MPR are dynamically constrained mechanisms employing a large number of on-off actuators that exert either a constant force (active state) or no force (inactive state) irrespective of their arbitrary kinematically unconstrained configuration. To achieve high force capabilities (both in terms of variation range and accuracy), the architecture of these MPR practically requires a large number of actuators (typically 4-10 times greater than the number

V. Padois, P. Bidaud, O. Khatib (Eds.), *Romansy 19 – Robot Design, Dynamics and Control*, CISM International Centre for Mechanical Sciences, DOI 10.1007/978-3-7091-1379-0_6,

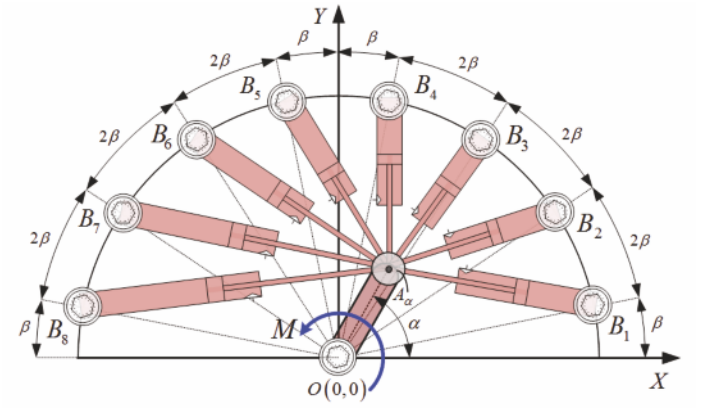

Figure 1. Ternary Massively Parallel Robot (MPR) actuated by eight three-state force generators.

of degrees of freedom desired for the robot) that are usually arranged in a prevalently in-parallel configuration. Owing to the large number and the discrete nature of the actuator variables, the Inverse Static Analysis (ISA) of MPR, i.e. to find the states of the actuator variables for a given external (force/torque) action, turns out to be a very challenging problem. To tackle this issue, in the last twenty years several elegant solution methods have been proposed [2-6], which however require too many calculations for on-line MPR control. In this paper, the potentialities of artificial intelligence methods are investigated for the real-time ISA solution of a planar ternary (i.e. with three states being -1, 0 and 1) MPR with one degree of rotational motion that is actuated by eight three-state force generators.

2 Ternary Massively Parallel Mechanism

The ternary MPR considered in this study is depicted in Fig. 1. It features eight identical Crank and Slotted-Lever (CSL) 3RP planar mechanisms (where R and P are for revolute and prismatic joints respectively) sharing the same crank and its moving revolute joint, which is centered at point $A(\alpha)$. The common crank is hinged to the base at point O, the eight links with variable length (hereafter called slotted-levers) $A(\alpha)B_i$ (for $i = 1, 2, \ldots, 8$) are hinged to the crank at the common point $A(\alpha)$ and to the base at points B_i, which are symmetrically located with respect to the Y axis along a circular arc having radius $r = OB_i$ and with spread angle 2β (here $\beta = 11.25°$). Possible implementation of the MPR mech-

anism depicted in Fig. 1 could be made by using cylindrical pneumatic or dielectric elastomer 3-way actuators. The output link of the considered MPR is the common crank. A discrete actuation is provided by the eight P joints through identical three-state force generators which, irrespective to the relative position of slider and slotted-lever, supply the forces

$$\mathbf{F}_i = F u_i \left[A\left(\alpha\right) - B_i\right] / \left\|A\left(\alpha\right) - B_i\right\|, \; i = 1, .., 8, \tag{1}$$

with F being a constant magnitude force and u_i being the activation state (namely $u_i \in \{1,\ 0,\ -1\}$) of the i-th actuator. Irrespective to the discrete actuation, the common crank can undergo continuous motion which is limited here in the range $0 \leq \alpha \leq 180°$. By considering all force contributions, the resulting torque M that can be generated at the output crank is

$$M\left(\alpha, u_i\right) = F \sum_{i=1}^{8} u_i \left[\mathbf{k} \cdot \left[A\left(\alpha\right) - O\right] \times \left[A\left(\alpha\right) - B_i\right] / \left\|A\left(\alpha\right) - B_i\right\|\right], \tag{2}$$

where $\boldsymbol{k}$ is the unit vector normal to the plane of motion of the mechanism. Equation (2) represents the static equilibrium condition of the considered ternary MPR (link weight and friction are ignored). For any desired continuous value α^D (i.e. for any desired MPR configuration), the Direct Static Analysis problem amounts to find the torque M^*, within a range $\mathcal{M}$ of discrete values, which corresponds to a known combination of the activation states u_i^D. Conversely, for any α^D, the Inverse Static Analysis (ISA) problem amounts to find the best combination of the activation sates u_i^* (among a total of 3^8 possibilities for each α^D) which enables the generation of the moment M^* (i.e. $M^* = M(\alpha^D,\ u_i^*)$) that more closely matches a desired torque M^D; that is, to find u_i^*, $i = 1, \ldots, 8$, for which the error $e^* = \left\|M^D - M^*\right\|$ is

$$e^* = \min_{u_i \in \{1,0,-1\}} \left(\left\|M^D - M\left(\alpha^D, u_i\right)\right\|\right). \tag{3}$$

Note that since the desired M^D can be any real value, whereas $\mathcal{M}$ is only a discrete subset, in general the minimum error e^* is different than zero. Moreover, owing to the discrete nature of the eight variables u_i, the ISA described by Eq. (3) cannot be solved via standard pseudo-inverse methods. To give an idea of the potential performances of the considered ternary MPR, the ranges $\mathcal{M}$ of available torques that can be generated at the output crank by discretely activating the eight actuators u_i are shown in Figs. 2 and 3 with cyan dot marks (data are computed via Eq. (2) with $F = 10$N, $||A(\alpha) - O\ || = 0.1$m and $r = 0.38$m). In the plots, each line corresponds to

a different angular position of the crank (specifically with α ranging from 0 to 90° and from 15° to 85° with 10° step in Figures 2 and 3 respectively). As shown, despite the discrete activation, this MPR is capable of generating torques in a rather ample range and with a reasonable resolution. Additionally, owing to the possibility of spatially distributing the partitioned actuation system, this MPR also exhibits a rather uniform torque generation capability within its full range of motion ($0 \leq \alpha \leq 180°$). Note that this latter feature cannot be achieved by a standard CSL mechanism actuated by a single continuously regulated force generator.

3 Inverse Static Analysis Models

This section presents five different methods for the solution of the ISA problem described in the previous section, namely: one Look-Up Table model; two Neuro-Fuzzy models; two Neural Network models. Essentially, each of these models is a computational machine that associates an output ternary number $\boldsymbol{u} = [u_1, \ldots, u_8]$ with eight trits (ternary digits) to an input couple of continuum real numbers $\boldsymbol{X} = [X_1, X_2] = [\alpha, M]$. Set-up of all these methods requires the knowledge of an appropriate input-output ($\boldsymbol{X}$-$\boldsymbol{u}$) dataset $\mathcal{D}$ with finite dimensions. Here, $\mathcal{D}$ consists of $10{\cdot}3^8$ $\boldsymbol{X}$-$\boldsymbol{u}$ correspondences that are generated via Eq. (2) for ten different values of α, ranging from 0 to 90° with 10° step, and for all possible (3^8) combinations of $\boldsymbol{u}$ (note that all the $\boldsymbol{X}$-$\boldsymbol{u}$ pairs contained in $\mathcal{D}$ satisfy Eq. (3) with $e^* = 0$). Given the continuity of α, $\mathcal{D}$ is not an exhaustive enumeration of all the possible solutions of the ISA problem. Thus the considered methods are required to provide some generalization ability (namely the ability to find $\boldsymbol{X}$-$\boldsymbol{u}$ pairs for arbitrary α which are not contained within $\mathcal{D}$). To discuss about their suitability for real-time control, the five ISA models are compared in terms of time of off-line preparation t_p, time of on-line calculation t_c, modeling error e_m (i.e. the error calculated via Eq. (3) in predicting $\boldsymbol{X}$-$\boldsymbol{u}$ correspondences for input pairs $\boldsymbol{X}^D = [\alpha^D, M^D]$ contained in $\mathcal{D}$), and generalization error e_g (i.e. the error calculated via Eq. (3) in predicting $\boldsymbol{X}$-$\boldsymbol{u}$ correspondences for input pairs $\boldsymbol{X}^D$ not contained in $\mathcal{D}$).

3.1 Look-Up Table Model

The Look-Up Table (LUT) model is a brute-force solution approach and it is the simplest method considered here. LUT uses a stored data structure as a pattern collection of the entire dataset $\mathcal{D}$ described above. As such, LUT does not require any learning procedure. During model preparation, the input values $\boldsymbol{X}$ of $\mathcal{D}$ are first normalized between 0 and 1, then the so

modified dataset $\mathcal{D}$ is sorted and stored row-by-row in an array. During model usage, the desired inputs $\boldsymbol{X}^D$ are first normalized, second they are compared to the corresponding entries of the LUT using a row-by-row similarity procedure, finally the suitable outputs u_i^* (for $i = 1, .., 8$) are chosen from the LUT row which provides the minimum error between $\boldsymbol{X}^D$ and $\boldsymbol{X}$.

3.2 Neuro-Fuzzy Models

Two Neuro-Fuzzy models are considered which are based on the Neuro-Fuzzy Takagi-Sugeno inference scheme with Gaussian membership functions [7]. Models of this kind are precise fuzzy systems which are static, easy to interpret, focus on accuracy and provide a strong connection between input $\boldsymbol{X}$ and output $\boldsymbol{u}$. Both models are based on the same overall architecture and only differ in the defuzzification operation. In particular, introducing the Gaussian membership functions G_j^n ($j = 1, 2$; $n = 1, \ldots, N$)

$$G_j^n\left(X_j\right) = \exp\left[-\left(X_j - c_j^n/\sigma_j^n\right)^2\right], \tag{4}$$

with characteristic mean c_j^n and variance σ_j^n, together with the fuzzy rules

$$R^n : \text{IF } X_1 \text{ is } G_1^n \text{ AND } X_2 \text{ is } G_2^n \text{ THEN } y_i^n = w_{0i}^n + w_{1i}^n X_1 + w_{2i}^n X_2, \tag{5}$$

with w_{0i}^n, w_{1i}^n and w_{2i}^n being the Takagi-Sugeno weights, the common part of the two Neuro-Fuzzy models calculates the continuous variables

$$\bar{u}_i = \sum_{n=1}^{N} y_i^n \left[\prod_{j=1}^{2} G_j^n\left(X_j\right) \Big/ \sum_{n=1}^{N}\prod_{j=1}^{2} G_j^n\left(X_j\right)\right]. \tag{6}$$

From Eq. (6), the two different models, hereafter briefly referred to as NFTS and NFLUT, are derived by alternatively estimating the actuator activation states u_i through one of the following defuzzification operations

$$u_i = round\left(\bar{u}_i\right) \text{ or } u_i = RLUT\left(\bar{u}_i\right) \tag{7}$$

where $RLUT$ indicates a properly predisposed Reduced Look-Up Table involving $\bar{u}_i$ as only input. Prior to their use, NFTS and NFLUT models require the tuning of the parameters c_j^n, σ_j^n, w_{0i}^n, w_{ji}^n (for $j = 1, 2$; $i = 1, \ldots, 8$; $n = 1, \ldots, N$; in the following $N = 11$). Here, the optimal values of these 308 parameters are found by a learning procedure which employs 12% of the $\boldsymbol{X}$-$\boldsymbol{u}$ pairs known from $\mathcal{D}$. In particular, the learning is performed via the Levenberg-Marquardt Algorithm [8]. Additionally, the NFLUT requires the generation of the RLUT, which is here constructed by storing the most significant $\boldsymbol{u}$-$\overline{\boldsymbol{u}}$ correspondences that occurred during the training with the known dataset $\mathcal{D}$.

3.3 Recurrent Neural Network Models

Two Neural Network models are considered which are based on Elman-type Recurrent Neural Networks with hyperbolic-tangent activation functions. Approximators of this kind are dynamic models that feature short-term memory so as to be capable of representing time-dependent mappings [9]. Both models are based on the same overall architecture and only differ in the presence or absence of the context layer. In particular, for a given input $\boldsymbol{X}(t) = [X_1(t),\, X_2(t)] = [\alpha(t),\, M(t)]$ at the time step t, both models calculate the actuator activation states $u_i(t)$ (for $i = 1, \ldots, 8$) as

$$u_i\,(t) = round\left[G\left(b_{2i} + \sum_{l=1}^{L} w_{il}^{HO} G\,(a_l\,(t))\right)\right] \text{ with } G\,(y) = y\Big/\sqrt{1+y^2}, \tag{8}$$

$$a_l\,(t) = b_{1l} + w^{CL} a_l\,(t-1) + \sum_{j=1}^{2} w_{lj}^{IH} X_j\,(t) \text{ with } a_l\,(0) = 0, \tag{9}$$

where b_{1l}, b_{2i}, w_{il}^{HO}, w_{lj}^{IH} and w^{CL} (for $i = 1, \ldots, 8$; $j = 1, 2$; $l = 1, \ldots, L$; in the following $L = 27$) are model parameters to be determined. From Eq. (8) and (9) the two different models, hereafter briefly referred to as MLP (Multi-Layer Perceptron) and ERNN, are derived by respectively selecting $w^{CL} = 0$ and $w^{CL} = 1/L$. Regarding the remaining 305 parameters, the optimal values are found by a learning procedure which employs 12% of the $\boldsymbol{X}$-$\boldsymbol{u}$ correspondences known from $\mathcal{D}$. In particular, the learning is performed here via an accelerated version of the Back-Propagation Algorithm [10].

3.4 Comparison of the Five Inverse Static Analysis Models

Comparison of the five considered models is reported in Figs. 2 and 3, as well as in Table 1. From these results, it can be concluded:

- LUT provides the worst generalization capabilities and the largest computational time during the on-line phase which makes it unsuited for real-time control applications.

Method/Description	LUT	NFTS	NFLUT	MLP	ERNN
Off-line preparation time t_p (s)	48	965	983	11358	639
On-line computation time t_c (s)	0.3502	0.0014	0.0135	0.0026	0.0026
Modeling error e_m (Nm)	0	0.658	0	0.464	0.377
Generalization error e_g (Nm)	0.805	0.985	0.528	0.515	0.389
Standard deviation of e_g (Nm)	0.499	0.585	0.328	0.431	0.347
Full Scale General. error (%)	16.1	19.7	10.5	10.3	7.8
Note: the CPU has 32 bit operating system, dual core processor, 2.6 GHz, RAM 4 GB.					

Table 1. Performance comparison of the considered methods.

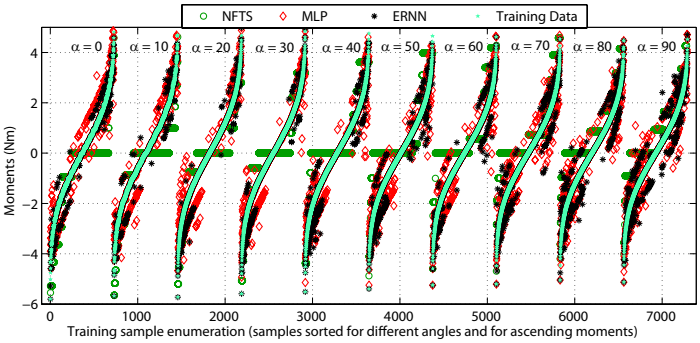

Figure 2. Training performance of different inverse static analysis methods.

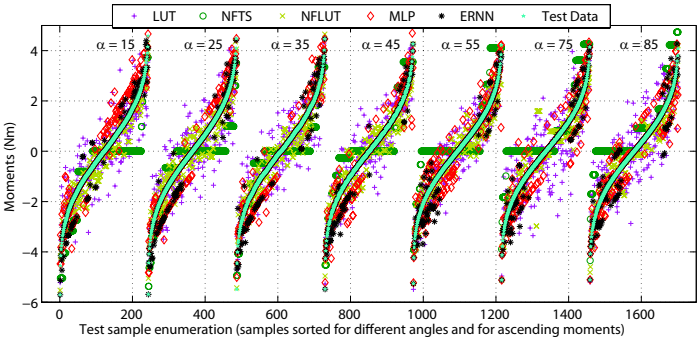

Figure 3. Testing performance of different inverse static analysis methods.

- ERNN is the most accurate model, features the best generalization ability, and requires a rather small computational time during the on-line phase. These features make ERNN very suited for real-time control.
- MLP is comparable to ERNN in terms of modeling accuracy, generalization capability and required on-line computational time. However, it needs very long time for off-line learning.
- NFTS features the shortest on-line computational time; however it is more inaccurate than MLP and ERNN both for modeling and for generalizing.
- NFLUT is rather similar to ERNN in terms of accuracy, but requires a larger on-line computational time.

4 Conclusions

This paper presented: 1) a planar massively parallel robot with 8 three-state force actuators and one continuous degree of rotational motion; 2) one

brute-force method, two Neuro-Fuzzy methods and two Recurrent Neural Network methods for the solution of the inverse static analysis. Thanks to the partitioned and spatially distributed actuator architecture, the considered discrete robot features rather ample, uniform and accurate torque generation capabilities. Comparison among the considered inverse static analysis methods highlighted that Elman type Recurrent Neural Network model is best suited for real-time control applications.

Bibliography

[1] B. Roth, J. Rastegar, and V. Sheinman, "On the design of computer controlled manipulators," in *First CISM-IFTOMM Symposium on Theory and Practice of Robots and Manipulators*, 1973, pp. 93–113.

[2] G. S. Chirikjian, "Inverse kinematics of binary manipulators using a continuum model," *J. of Intelligent and Robotic Systems*, vol. 19, pp. 5–22, 1997.

[3] I. Ebert-Uphoff and G. S. Chirikjian, "Inverse kinematics of discretely actuated hyper-redundant manipulators using workspace densities," in *IEEE Int. Conf. on Robotics and Automation*, 1996, pp. 139–245.

[4] J. Suthakorn and G. S. Chirikjian, "A new inverse kinematic algorithm for binary manipulators with many actuators," *Advanced Robotics*, vol. 15, no. 2, pp. 225–244, 2001.

[5] D. Lichter, V. A. Sujan, and S. Dubowsky, "Computational issues in the planning and kinematics of binary robots," in *IEEE Int. Conf. on Robotics and Automation*, 2000, pp. 341–346.

[6] P. Yang and K. J. Waldron, "Massively parallel actuation," in *IEEE/ASME Int. Conf. on Advanced Intelligent Mechatronics*, 2001, pp. 868–873.

[7] J. S. R. Jang, "Anfis: adaptive-network-based fuzzy inference system," *IEEE Trans. on Systems, Man and Cybernetics*, vol. 23, no. 3, pp. 665 –685, 1993.

[8] A. Palit and R. Babuska, "Efficient training algorithm for takagi-sugeno type neuro-fuzzy network," in *IEEE Int. Conf. on Fuzzy Systems*, vol. 3, 2001, pp. 1367–1371.

[9] J. Elman, "Finding structure in time," *Cognitive Science*, vol. 14, pp. 179–211, 1990.

[10] A. K. Palit and D. Popovic, *Computational Intelligence in Time Series Forecasting, Theory and Engineering Applications*. Springer, 2005.