Data-Driven Multi-Stage Motion Planning of Parallel Kinematic Machines

Amar Khoukhi

***Abstract*—A multistage data-driven neuro-fuzzy system is considered for the multiobjective trajectory planning of Parallel Kinematic Machines (PKMs). This system is developed in two major steps. First, an offline planning based on robot kinematic and dynamic models, including actuators, is performed to generate a large dataset of trajectories, covering most of the robot workspace and minimizing time and energy, while avoiding singularities and limits on joint angles, rates, accelerations, and torques. An augmented Lagrangian technique is implemented on a decoupled form of the PKM dynamics in order to solve the resulting nonlinear constrained optimal control problem. Then, the outcomes of the offline-planning are used to build a data-driven neuro-fuzzy inference system to learn and capture the desired dynamic behavior of the PKM. Once this system is optimized, it is used to achieve near-optimal online planning with a reasonable time complexity. Simulations proving the effectiveness of this approach on a 2-degrees-of-freedom planar PKM are given and discussed.**

***Index Terms*—Augmented Lagrangian, data-driven neuro-fuzzy systems, decoupling, multiobjective trajectory planning, parallel kinematic machines, subtractive clustering.**

I. Introduction

SINCE the pioneering work of Gough [1] on manipulators with closed-loop structures, extensive research efforts have led to the realization of several robots and machine tools with parallel kinematic structures, with a significant diversity in design, specification, and personalization [2]–[5]. Parallel kinematic machines (PKMs) have two basic advantages over conventional machines of serial kinematic structures. First, with PKM structures, it is possible to mount all drives on or near the base. This yields to large payloads capability and low inertia. Indeed, the ratio of payload to the robot load is usually about 1/10 for serial robots, while only 1/2 for parallel ones. Second, the connection between the base and the end-effector (EE) is made with several kinematic chains. This results in high structural stiffness and rigidity. However, the PKM architecture-dependent performance associated with strong-coupled nonlinear dynamics makes the trajectory planning and control system design for PKMs more difficult compared to serial machines [2]–[4]. Another major issue for practical use of PKMs in industry is that for a prescribed tool path in the workspace, the control system should guarantee the prescribed task completion within the workspace, for a given set up of the EE (i.e., for which limitations on actuator lengths and physical dimensions are not violated). In [5] a particle swarm optimization technique was used to plan a singularity-free minimum-effort trajectory of a PKM. In [6], methodologies involving workspace and actuator force limitations were proposed, using iterative optimization techniques. However, these methods did not include multiobjective nature of the problem and EE passing through imposed positions and orientation constraints. An integrated multiobjective dynamic offline trajectory planning system for PKMs is given in [7] and it had shown very good results. One of the major online motion planning algorithm problems is that a desired trajectory may cause saturation of the speed and/or torques delivered by the joint actuators in the vicinity of singularities or in any other region of the workspace due to nonlinear kinematic transformations between task and joint spaces. Moreover, if the planned trajectory was calculated on a minimum-time basis, it had been shown that the resulting tracked trajectory will be instable and may lead to saturation of the speed and/or torques. As a matter of fact, conventional fixed gain, linear feedback controllers are not capable of effectively controlling the movements of multijoint robot manipulators under different distance, velocity and load requirements. Nonlinear feedback approaches like computed torque and H_∞ provide better compensation for the dynamic interactions present in various robot motions [8]. However, these approaches require complete nonlinear dynamic models, which are difficult to be accurately implemented in real-time. Recently, soft computing had been seen as an attractive alternative and several methods were developed for trajectory design and robot motion control using neuro-fuzzy techniques [9]–[13]. They are computationally efficient after training and had shown very good learning and generalization capabilities if a large enough dataset points representing the system's behavior is given. In this context, this brief develops a data-driven multistage multiobjective motion planning for PKMs. The best trajectories can be obtained by a constrained optimal control based on non linear kinematic and dynamic models and constraints relating robot, workspace, and task interactions (see Fig. 1). An offline multiobjective planning is performed using an augmented Lagrangian technique implemented on a decoupled form of the PKM dynamics. This step is done as many times as possible to cover mostly of the robot workspace. The generated trajectories are gathered into an input/output dataset. The inputs are the EE positions and velocities, and the outputs are the joint actuator torques and sampling periods. Then an adaptive neuro-fuzzy system, named neuro-fuzzy multiobjective planning (NeFuMOP) is built. It starts with a subtractive clustering, allowing initialization of membership function parameters and number of rules of the neuro-fuzzy system. Then, NeFuMOP is optimized in order to learn and capture the dynamic multiobjective behavior of the PKM. Once the neuro-fuzzy structure (rules number and premise and

Manuscript received May 21, 2009; revised August 11, 2009. Manuscript received in final form November 05, 2009. First published December 22, 2009; current version published October 22, 2010. Recommended by Associate Editor C.-Y. Su. This work was supported by King Fahd University of Petroleum and Minerals.

The author is with the Department of System Engineering, King Fahd University of Petroleum and Minerals, Dhahran 31261, Saudi Arabia (e-mail: amar@kfupm.edu.sa).

Color versions of one or more of the figures in this brief are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCST.2009.2036600

1063-6536/$26.00 © 2009 IEEE

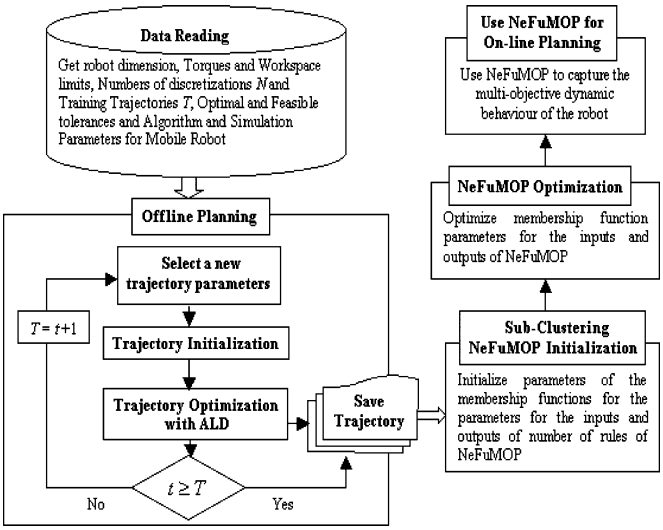

Fig. 1. Overall diagram of the proposed approach.

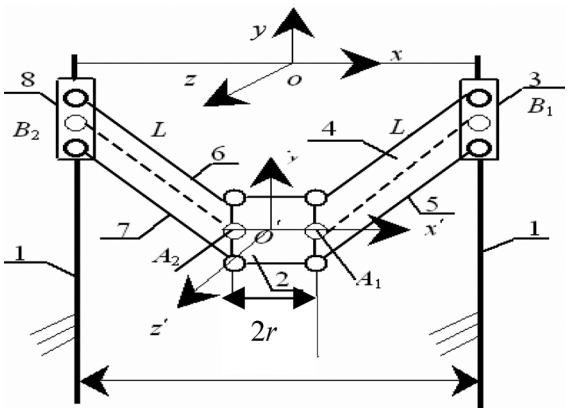

Fig. 2. Geometric representation of the PKM.

consequence membership function parameters) is identified and optimized, it is used in a generalization phase to achieve near-optimal online planning with a reasonable computational complexity. Near-optimality steams from the fact that at the generalization level, NeFuMOP interpolates within the parts of the domain that are not covered by offline planning dataset. Section II is devoted to offline data set building. In Section III, we introduce NeFuMOP. In Section IV, an implementation on a 2-DOF planar PKM is provided and Section V concludes this work.

II. Offline Planning and Dataset Building

A. Kinematic Model

The kinematic model is briefed in this section. More details on the modeling and augmented Lagrangian technique are provided in [7] and [14]. The planar PKM shown in Fig. 2 represents a 2-DOF motion of its EE through articulated motion of its two leg lengths. In this PKM, the base is labeled 1 and the EE is labeled 2. The EE is connected to the base by two identical legs. Each leg consists of a planar four-bar parallelogram: links 2, 3, 4, and 5 for the first leg and links 2, 6, 7, and 8 for the second leg. Prismatic actuators actuate the links 3 and 8, respectively. Motions of the EE are achieved by combination of movements of links 3 and 8 that can be transmitted to the EE by the system of the two parallelograms. Due to its structure, the manipulator can position a rigid body in 2-D space with a constant orientation. To characterize the planar four-bar parallelogram, the chains A_1B_1 and A_2B_2 are considered as shown in Fig. 2. Vectors $A_{i\boldsymbol{A}}$ and $A_{i\boldsymbol{B}}$ $(i = 1, 2)$ define the positions of points A_i in frames A and B, respectively. Vectors $B_{i\boldsymbol{B}}$ $(i = 1, 2)$ define the position of B_i points in frame B. The geometric parameters are

$$A_iB_i = L \quad (i = 1, 2) \tag{1}$$
$$A_1A_2 = 2r \quad B_1B_2 = 2R. \tag{2}$$

The closure of each kinematic loop passing through the origin of frame A and frame B, and through attachment points B_i on the base and the hip attachment points A_i on the EE is given as

$$\boldsymbol{J}_l \begin{bmatrix} \dot{y}_1 \\ \dot{y}_2 \end{bmatrix} = \boldsymbol{J}_x \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix}$$
$$\boldsymbol{J}_l = \begin{bmatrix} y - y_1 & 0 \\ 0 & y - y_2 \end{bmatrix}$$
$$\boldsymbol{J}_x = \begin{bmatrix} r + x + R & y - y_1 \\ x - r + R & y - y_2 \end{bmatrix} \tag{3}$$

where $[y_1 y_2]^T$ is the position vector of actuated points B_1 and B_2, $[xy]^T$ defines the position of point O' in the fixed frame B, $\boldsymbol{J}_l$ and $\boldsymbol{J}_x$ are the 2×2 inverse and forward Jacobian; If $\boldsymbol{J}_l$ is nonsingular, the PKM's Jacobian is

$$\boldsymbol{J} = \boldsymbol{J}_l^{-1}\boldsymbol{J}_x = \begin{bmatrix} (r + x - R)/(y - y_1) & 1 \\ (x - r + R)/(y - y_2) & 1 \end{bmatrix} = \begin{bmatrix} J_{11} & J_{12} \\ J_{21} & J_{22} \end{bmatrix}. \tag{4}$$

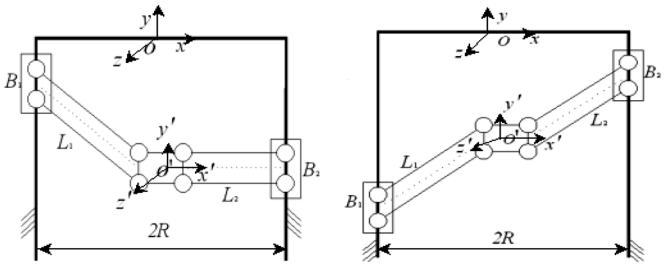

Fig. 3. Type-1 and Type-2 singularities of the PKM.

Hence, it is clear that singularity occurs in the following cases.

First Case: $|\boldsymbol{J}_l| = 0$ and $|\boldsymbol{J}_x| \neq 0$. This corresponds to a type-1 singularity and the situation where $y = y_1$ or $y = y_2$, i.e., the first or the second leg is parallel to the x-axis (see Fig. 3).

Second Case: $|\boldsymbol{J}_l| \neq 0$ and $|\boldsymbol{J}_x| = 0$. This is a type-2 singularity where a pose for four bars of the parallelogram in one of the two legs are parallel to each other, (e.g., $x + r = R$ for the first leg for $x \geq 0$, and $x + R = r$ for the second leg for $x \leq 0$).

Third Case: $|\boldsymbol{J}_l| = 0$ and $|\boldsymbol{J}_x| = 0$. This is a type-3 singularity for which the two legs are both parallel to the x-axis.

This is characterized by a geometric parameter condition given by

$$L + r = R. \tag{5}$$

These parameters are designed, such that (5) never holds.

B. Dynamic Model

In the offline planning system, one can include contact efforts. Among such models, there are friction and other application-specific forces. This can increase task achievement success in many practical applications like grinding, or screwing. It allows avoiding actuator saturation and improving the trajectory planning performance. One way to do so is by adding these efforts as disturbance inputs to the dynamic equations. The joint space inverse dynamic model including contact forces is given as [2]–[5], [7]

$$\boldsymbol{\tau} = \boldsymbol{M}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{N}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{G}(\boldsymbol{q}) + \boldsymbol{J}^T \boldsymbol{f}_c(\dot{\boldsymbol{q}}) \tag{6}$$

where $\boldsymbol{\tau}$ is the torques vector produced by joint actuators, $\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}$ are joint positions, rates, and accelerations, $\boldsymbol{M}(\boldsymbol{q})$ is the inertia matrix, $\boldsymbol{N}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ and $\boldsymbol{G}(\boldsymbol{q})$ are the Coriolis and centrifugal, and gravitational forces, respectively, $\boldsymbol{J}$ is the PKM Jacobian and $\boldsymbol{f}_c$ is the contact force. The model in (6) is represented as follows:

$$\ddot{\boldsymbol{q}} = \boldsymbol{M}^{-1}(\boldsymbol{q}) \left[\boldsymbol{\tau} - \boldsymbol{N}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} - \boldsymbol{G}(\boldsymbol{q}) - \boldsymbol{J}^T \boldsymbol{f}_c(\dot{\boldsymbol{q}}) \right]. \tag{7}$$

By defining $\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2)^T = (\boldsymbol{q}, \dot{\boldsymbol{q}})^T$, (7) is rewritten as

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\tau}, h) \tag{8}$$

where $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ denote the 2-D joint positions and velocities, $\boldsymbol{f}$ describes the dynamics of the PKM.

C. Constraints Modeling

In addition to (8), one has the following constraints.

- *Sampling Period Limits:*

$$h_{\mathrm{Min}} \leq h \leq h_{\mathrm{Max}}. \tag{9}$$

- *Boundary Conditions:*

$$x_0 = x_s, \quad x_N = x_T. \tag{10}$$

- *Actuator Torque Limits:*

$$\boldsymbol{\tau}_{\mathrm{Min}} \leq \boldsymbol{\tau}_k \leq \boldsymbol{\tau}_{\mathrm{Max}}. \tag{11}$$

- *Workspace Limitations:*

$$x_{\mathrm{Min}} \leq x_k \leq x_{\mathrm{Max}},\ y_{\mathrm{Min}} \leq y_k \leq y_{\mathrm{Max}},\ k = 0, 2, \ldots, N. \tag{12}$$

- *Singularity Avoidance:* Because the robot Jacobian allows motion and force transformation from actuated legs to the EE, the leg forces demand at a given point on the trajectory must be continuously checked for possible violation of the preset limits as the manipulator moves close to singularity. The condition number of the Jacobian was suggested and used as a local performance index for evaluating the velocity, accuracy, and rigidity mapping characteristics between the joint variables and the moving platform [2]–[7], [15]

$$\kappa_{\mathrm{Min}} \leq \mathrm{Cond}\left(J(x_k)\right) \leq \kappa_{\mathrm{Max}} \tag{13}$$

where κ_{Min} and κ_{Max} correspond, respectively, to the minimum and maximum values for the tolerances on the condition number. Appendix A provides an expression of the condition number for the PKM at hand.

- *Imposed Passages:* These constraints are quantified by a set of L poses with $\mathbf{p}_l$ referring to the l^{th} imposed Cartesian position on the EE

$$\|\mathbf{p} - \mathbf{p}_l\| \leq \mathbf{T}_{\mathbf{PassTh}lp}, \quad l = 1, \ldots, L \tag{14}$$

where $\mathbf{T}_{\mathbf{PassTh}lp}$ is the passage tolerance
For writing simplicity, all equality constraints are given as

$$\boldsymbol{s}(\boldsymbol{x}) = \mathbf{0}, \quad i = 1, \ldots, \mathbf{I} \tag{15}$$

and all inequality constraints are noted as

$$\mathbf{g}_j(\boldsymbol{x}, \boldsymbol{\tau}, h) \leq \mathbf{0}, \quad j = 1, \ldots, \mathbf{J} \tag{16}$$

regardless whether they depend only on the state, control variables, or both.

D. Performance Index

The discrete-time optimal control problem can be stated as: *among all admissible control sequences* $(\boldsymbol{\tau}_0, \boldsymbol{\tau}_2, \ldots, \boldsymbol{\tau}_{N-1}) \in \mathrm{C}$ *and* $h \in \mathrm{H}$, *that allow the robot to move from an initial state* $\boldsymbol{x}_0 = \boldsymbol{x}_S$ *to a final state* $\boldsymbol{x}_N = \boldsymbol{x}_T$, *find those that minimize the cost function* E_d:

$$\underset{\substack{\boldsymbol{\tau} \in \mathrm{C} \\ h \in \mathrm{H}}}{\mathrm{Min}} E_d = \left\{ \sum_{k=0}^{N-1} \left[\boldsymbol{\tau}_k \boldsymbol{U} \boldsymbol{\tau}_k^T + \boldsymbol{\iota}_1 + \boldsymbol{x}_{2k} \boldsymbol{Q} \boldsymbol{x}_{2k}^T \right] h_k + \frac{1}{2} \boldsymbol{x}_{2N} \boldsymbol{Q} \boldsymbol{x}_{2N}^T \right\}. \tag{17}$$

Subject to constraints (8)–(14), where C and H represent the sets of admissible torques and sampling periods and $\boldsymbol{U}$, $\boldsymbol{Q}$ and $\boldsymbol{\iota}$ are, the weight factors of the electric energy, kinetic energy, and travel time.

E. Augmented Lagrangian With Decoupling (ALD)

There are two basic approaches to solve the stated multiobjective nonlinear optimal control problem (17); the dynamic programming through global optimal control and variational calculus through the maximum principle. With the former approach, an optimal feedback control $\boldsymbol{\tau}^*(\boldsymbol{x}, t)$, can be characterized by solving for a so-called value function through Hamilton–Jacobi–Bellman partial differential equations [16]. For a general system however, the PDE can be solved numerically for very small state dimension only. If inequality

constraints on state and control variables are added, this makes the problem even harder. We propose to use the second approach [17]. An augmented Lagrangian (AL) technique is implemented to solve the multiobjective nonlinear optimal control problem (17). This technique transforms the constrained problem into a nonconstrained one, where the degree of penalty for violating the constraints is regulated by penalty parameters [17]. Moreover, while ordinary Lagrangian methods are used when the objective function and the constraints are convex, for the case at hand, we cannot tell whether these are convex or not, but they are most likely not. The AL technique relies on quadratic penalty methods but reduces the possibility of ill conditioning of the sub-problems that are generated with penalization by introducing explicit Lagrange multipliers estimates at each step into the function to be minimized. However, in developing the first order optimality conditions enabling one to derive the iterative formulas to solve the optimal control problem, in (8), $\boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{\tau}_k, h_k)$ contains the inverse of the total inertia matrix $\boldsymbol{M}^{-1}(\boldsymbol{x})$ of the PKM, including struts and actuators, as well as their Coriolis and centrifugal wrenches $\boldsymbol{N}(\boldsymbol{x}_1, \boldsymbol{x}_2)$. These would take several pages long to display. In computing the adjoin states $\boldsymbol{\lambda}_k$, one has to determine the inverse of the mentioned inertia matrix and its derivatives with respect to state variables, resulting in an intractable complexity. This major computational difficulty is solved using a linear-decoupled formulation [18].

Theorem 1: Under the invisibility condition of the inertia matrix, the control law defined in the Cartesian space as

$$\boldsymbol{u}=\boldsymbol{M}\boldsymbol{J}^{-1}\boldsymbol{v}+\boldsymbol{N}\boldsymbol{x}_2+\boldsymbol{G}+[\boldsymbol{J}^T-\boldsymbol{M}\boldsymbol{J}^{-1}\boldsymbol{C}]\boldsymbol{f}_c-\boldsymbol{M}\boldsymbol{J}^{-1}\dot{\boldsymbol{J}}\boldsymbol{x}_2 \quad (18)$$

allows the robot to have a linear and decoupled behavior with the following dynamic equation, given in the task space:

$$\ddot{\mathrm{x}} = \boldsymbol{v} - \boldsymbol{C}\boldsymbol{f}_c. \quad (19)$$

Proof: See Appendix B.

For writing convenience, the second-order system of the decoupled dynamics (19) will be rewritten as

$$\dot{\boldsymbol{x}} = (\dot{\mathrm{x}}, \boldsymbol{v} - \boldsymbol{C}\boldsymbol{f}_c) = \boldsymbol{f}_c(\boldsymbol{x}, \boldsymbol{v}). \quad (20)$$

Because the ordinary differential equation dynamic governing the robot behavior is of a stiff type [19], a multistep Adams predictive-corrective technique is used to approximate the discrete dynamic model. This is initialized with a fourth order Runge–Kutta [19]. The resulting approximated discrete decoupled dynamics will be written as

$$\boldsymbol{x}_{k+1} = \boldsymbol{f}_k^D(\boldsymbol{x}_k, \boldsymbol{v}_k, h). \quad (21)$$

Remark: It is note worthy that the decoupled technique alleviates the need of calculating the inertia matrix inverse and its derivatives with respect to state variables at each iteration. However, the nonlinearity of the initial problem is not removed neither reduced. It is only transferred to the objective function. The decoupling transforms the discrete optimal control problem into finding optimal sequences of sampling periods and acceleration inputs $(h, \boldsymbol{v}_0, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_{N-1})$, allowing the robot to move from an initial state $\boldsymbol{x}_0 = \boldsymbol{x}_S$ to a final state $\boldsymbol{x}_N = \boldsymbol{x}_T$, while minimizing the cost function and satisfying the above mentioned constraints. The augmented Lagrangian with decoupling (ALD) is

$$\begin{aligned} L_\mu^{\mathrm{D}}&(\boldsymbol{x}_k, \boldsymbol{\nu}_k, h_k, \boldsymbol{\lambda}_k, \boldsymbol{\rho}_k, \boldsymbol{\sigma}_k) \\ &= \sum_{k=0}^{N-1} h_k \Big[\iota + \boldsymbol{x}_{2k}\boldsymbol{Q}\boldsymbol{x}_{2k}^T [\boldsymbol{M}(\boldsymbol{x}_{1k})\boldsymbol{v}_k + \boldsymbol{N}(\boldsymbol{x}_{1k}, \boldsymbol{x}_{2k}) \\ &\qquad + \boldsymbol{G}(\boldsymbol{x}_{1k}) + \boldsymbol{J}^T(\boldsymbol{x}_{1k})\boldsymbol{f}_c(\boldsymbol{x}_{2k})\Big]^T \\ &\qquad \times \boldsymbol{U}\,[\boldsymbol{M}(\boldsymbol{x}_{1k})\boldsymbol{v}_k + \boldsymbol{N}(\boldsymbol{x}_{1k}, \boldsymbol{x}_{2k}) + \boldsymbol{G}(\boldsymbol{x}_{1k}) \\ &\qquad + \boldsymbol{J}^T(\boldsymbol{x}_{1k}) f_c(\boldsymbol{x}_{2k})\Big]\Big] \\ &\quad + \sum_{k=0}^{N-1} \boldsymbol{\lambda}_{k+1}^T \Big[\boldsymbol{x}_{k+1} - \boldsymbol{f}_k^D(\boldsymbol{x}_k, \boldsymbol{\nu}_k, h_k)\Big] \\ &\quad + \sum_{k=0}^{N-1} h_k \Bigg[\sum_{j=1}^{2} \Phi_{\boldsymbol{\mu}_g}\Big(\boldsymbol{\rho}_k^j, \boldsymbol{g}_j(\boldsymbol{x}_k, \boldsymbol{v}_k)\Big) \\ &\qquad + \Psi_{\boldsymbol{\mu}_s}\left(\boldsymbol{\sigma}_k^i, \boldsymbol{s}_i(\boldsymbol{x}_k)\right)\Bigg] \end{aligned} \quad (22)$$

where $\boldsymbol{f}_{d_k}^D(\boldsymbol{x}_k, \boldsymbol{v}_k, h_k)$ is defined by the decoupled state (21), N is the total sampling number, $\boldsymbol{\lambda} \in R^{4\mathrm{N}}$ designates the costates, $\boldsymbol{\sigma}$ and $\boldsymbol{\rho}$ are Lagrange multipliers, associated to equality and inequality constraints and $\boldsymbol{\mu}_S$ and $\boldsymbol{\mu}_g$ are the penalty coefficients. The used penalty functions allow relaxation of inequality constraints as soon as they are satisfied. These functions are [17]

$$\begin{aligned} &\Psi_{\boldsymbol{\mu}_s}(\boldsymbol{a}, \boldsymbol{b}) = \left(\boldsymbol{a} + \frac{\boldsymbol{\mu}_s}{2}\boldsymbol{b}\right)^T \boldsymbol{b}, \text{ and} \\ &\Phi_{\boldsymbol{\mu}_g}(\boldsymbol{a}, \boldsymbol{b}) = \frac{1}{2\boldsymbol{\mu}_g}\left\{\left\|\mathbf{Max}(0, \boldsymbol{a} + \boldsymbol{\mu}_g\boldsymbol{b})\right\|^2 - \|\boldsymbol{a}\|^2\right\}. \end{aligned} \quad (23)$$

The Karush–Kuhn–Tucker first-order optimality conditions state that for a trajectory $(\boldsymbol{x}_0, \boldsymbol{\nu}_0, h_0, \ldots, \boldsymbol{x}_N, \boldsymbol{\nu}_{N-1}, h_{N-1})$, to be optimal solution to the problem, there must exists some positive Lagrange multipliers $(\boldsymbol{\lambda}_k, \boldsymbol{\rho}_k)$, unrestricted sign multipliers $\boldsymbol{\sigma}_k$ and finite positive penalty coefficients $\boldsymbol{\mu} = (\boldsymbol{\mu}_S, \boldsymbol{\mu}_g)$ such that

$$\begin{aligned} &\frac{\partial L_\mu^D}{\partial \boldsymbol{x}_k}=0,\ \frac{\partial L_\mu^D}{\partial \boldsymbol{\nu}_k}=0,\ \frac{\partial L_\mu^D}{\partial h_k}=0,\ \frac{\partial L_\mu^D}{\partial \boldsymbol{\lambda}_k}=0,\ \frac{\partial L_\mu^D}{\partial \boldsymbol{\rho}_k}=0 \\ &\frac{\partial L_\mu^D}{\partial \boldsymbol{\sigma}_k}=0,\ \boldsymbol{\sigma}_k^T\boldsymbol{s}(x)=0,\ \boldsymbol{\rho}_k^T\boldsymbol{g}(\boldsymbol{x}, \boldsymbol{\nu}, h)=0 \quad \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{\nu}, h)\leq 0. \end{aligned} \quad (24)$$

One notices that the final state constraint $\boldsymbol{x}_N = \boldsymbol{x}_T$ does not appear in the ALD function (22). This implies that the backward adjoin states integration will starts off with $\boldsymbol{\lambda}_N = 0$.

Because we seek to satisfy this constraint at each iteration to enhance the task execution precision, we treat it through a gradient projection. A readjustment is performed with an orthog-

onal projection on the tangent space of this constraint, through the application of a descent direction given as:

Theorem 2: The descent direction is expressed as

$$\boldsymbol{d} = -\mathbf{P}_{\boldsymbol{\nu}} \nabla_{\boldsymbol{\nu}} L_{\mu}^{D}. \tag{25}$$

With $\mathbf{P}$ being defined as

$$\mathbf{P} = \mathbf{I}_d - \mathbf{S}^T(\mathbf{S}\mathbf{S}^T)^{-1}\mathbf{S} \tag{26}$$

where $\mathbf{I}_d$ is an identity matrix with appropriate dimension and $\mathbf{S}$ is the projection matrix on the tangent space of the final state constraint. The re-adjustment process allows satisfying target attainability with any given ε-precision [17]. After initialization through a trapezoidal velocity profile [7], an inner optimization loop solves for the ALD minimization with respect to sampling periods and acceleration variables. The adjoin states are computed backwardly and the control inputs and the states are updated. All equality and inequality constraints are tested against feasibility tolerances. If nonfeasibility holds, the inner optimization unit is started over. If feasibility occurs, i.e., the current penalty values maintain good near-feasibility, a convergence test is made against optimal tolerances. If convergence holds, the optimal results are displayed and the program stops. If nonconvergence occurs, we go further into the dual part of ALD to update Lagrange multipliers, penalty, step size, and tolerances in order to force the subsequent iterates to generate increasingly accurate solutions to the primal problem.

III. Neuro-Fuzzy Multiobjective Planning

A. NeFuMOP Structure

The outcomes of the so-developed offline trajectory planning system are used to generate an input/output dataset on which to build a neuro-fuzzy multiobjective planner. This neuro-fuzzy system was implemented on serial manipulator and mobile robots [20] and gave very good results. It is described briefly hereafter. NeFuMOP is a data-driven neuro-fuzzy system based on Tsukamoto fuzzy inference mechanism. In this mode of reasoning, the consequence linguistic terms are assumed to have a continuous strong monotone membership function [21]. The inputs of the MIMO (multiple input multiple output) fuzzy model are the discrete Cartesian 2-D positions and velocities; $\mathrm{x}_d = [x_i, y_i, \dot{x}_i, \dot{y}_i] i = 0, \ldots, N$. The outputs are the joint torques and sampling periods given as $\boldsymbol{\pi}_d = [\boldsymbol{\pi}_1, \boldsymbol{\pi}_2, \ldots, \boldsymbol{\pi}_N]^T$, $\boldsymbol{\pi}_i = [\boldsymbol{\tau}_i, h]^T$, with $\boldsymbol{\tau}_i$ being the vector of joint torques and h the sampling period. Consider the following input/output entries $\lfloor (\mathbf{x}_d^1, \boldsymbol{\pi}_d^1), \ldots, (\mathbf{x}_d^K, \boldsymbol{\pi}_d^K) \rfloor$. The fuzzy rules are built as follows:

R^1 : If $\left(x_1 \text{ is } A_1^1\right)$ and $\left(x_2 \text{ is } A_2^1\right)$ and $\ldots$ and $\left(x_4 \text{ is } A_4^1\right)$,
then $\left(o_1 = c_1^1\right)$ and $\left(o_2 = c_2^1\right)$ and $\left(o_3 = c_3^1\right)$
also $\ldots$
R^j : If $\left(x_1 \text{ is } A_1^j\right)$ and $\left(x_2 \text{ is } A_2^j\right)$ and $\ldots$ and $\left(x_4 \text{ is } A_4^j\right)$,
then $\left(o_1 = c_1^j\right)$ and $\left(o_2 = c_2^j\right)$ and $\left(o_3 = c_3^j\right)$
$\ldots$
R^J : If $\left(x_1 \text{ is } A_1^J\right)$ and $\left(x_2 \text{ is } A_2^J\right)$ and $\ldots$ and $\left(x_4 \text{ is } A_4^J\right)$,
then $\left(o_1 = c_1^J\right)$ and $\left(o_2 = c_2^J\right)$ and $\left(o_3 = c_3^J\right)$

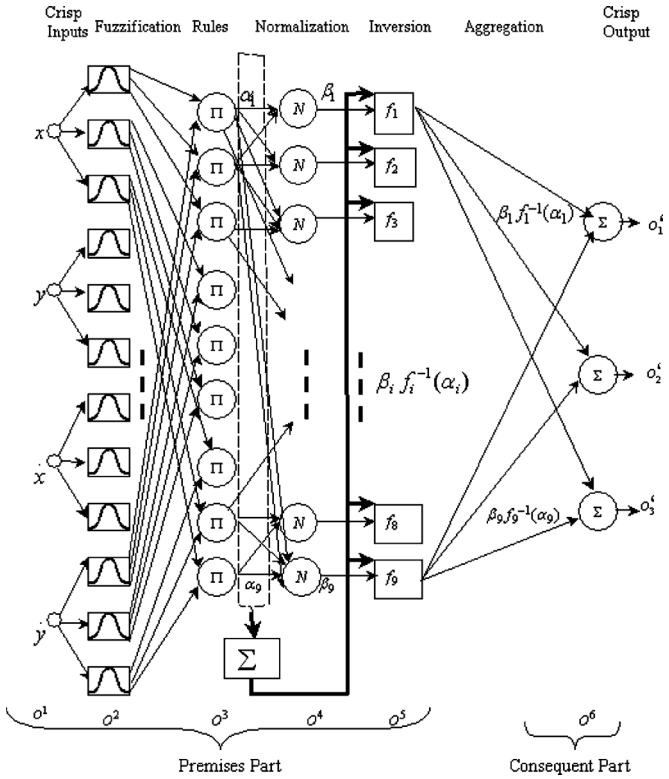

Fig. 4. NeFuMOP architecture.

where x_l is the input associated to node l, A_l^j is the associated linguistic term. The membership functions $\mu_{A_l^j}$ defining the fuzzy sets A_l^j

$$\mu_{A_l^j}(x_l) = \exp \frac{-1}{2} \left\{ \frac{\left(x_l - a_l^j\right)^2}{\left(b_l^j\right)^2} \right\},\ l = 1, \ldots 4,\ j = 1, \ldots, J \tag{27}$$

where a_l^j and b_l^j are the mean and standard deviation of the j^{th} membership function of the input variable x_l, and c_t^j are fuzzy sets defining the consequence of the j^{th} rule, such that

$$c_t^j = f^{-1}\left(\mu_{A_l^j}(x_l)\right),\ f : \Re \rightarrow [0,1],\ t = 1, \ldots, 3,\ j = 1, \ldots, J \tag{28}$$

f being a continuous strong monotone function defined as

$$f(z) = \frac{1}{1 + e^{-(\rho z - \sigma)}} \tag{29}$$

where ρ and σ are real numbers affecting the position and slope of the inflection point of f.

B. NeFuMOP Architecture and Learning

NeFuMOP is made of six layers (see Fig. 4). The first layer is a four-input layer, characterizing the crisp EE position and velocity. The second layer performs the fuzzification of the crisp inputs into linguistic variables, through Gaussian transfer functions. The third one is the rule layer, which applies the product t-norm to produce the firing strengths of each rule. This is followed by a normalization layer, at which each node calculates the ratio of a rule's firing strength to the sum of all rules firing

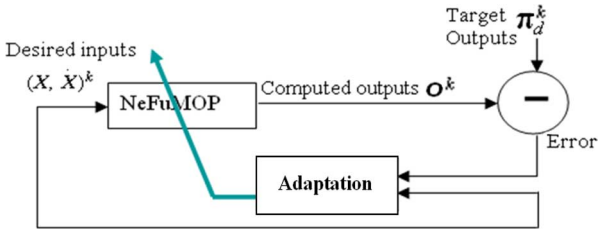

Fig. 5. NeFuMOP function and operation.

TABLE I
WORKSPACE, ACTUATOR TORQUES, AND SAMPLING PERIODS LIMITS

Parameter	x-coordinate (m)	y-coordinate (m)	τ_1 (Nm)	τ_2 (Nm)	h (sec)
max	0.8	-0.720	550	700	0.7
min	-0.8	-1.720	-550	-700	0.005

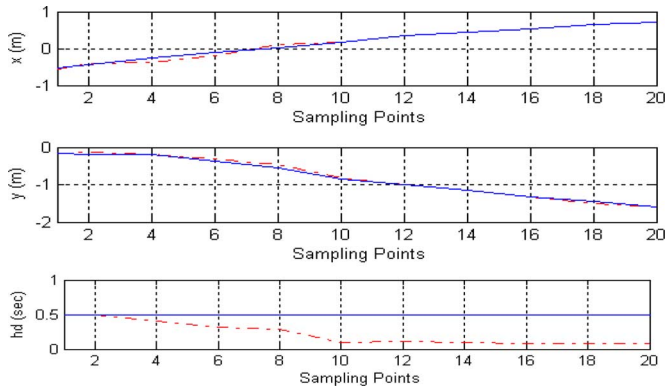

Fig. 6. Initial and ALD trajectories.

strengths. The fifth layer performs the inversion. The last layer is for aggregation and defuzification. The output is obtained as the sum of all incoming signals.

The learning scheme of NeFuMOP is composed of two steps: structure identification where the network parameters of the premise membership function parameters a_l, b_l, and the consequent parameters ρ_t, σ_t of a fuzzy rule R^j are initialized by partitioning the input/output dataset into clusters. The purpose of clustering is to identify natural grouping of data from the generated large data set to produce a concise representation of the PKM's behavior, resulting in initial rules that are more tailored to the input data. One assumes that it is not clear how many clusters there should be for the generated offline planning dataset. The subtractive clustering technique [21], [22] is applied to find the number of clusters and their centers. This technique provides a fast one-pass algorithm to take input/output training data to generate a fuzzy inference system that captures the robot dynamic behavior. Each cluster center may be translated into a fuzzy rule for identifying a class. The Matlab function subclust is used to initialize and identify these parameters. The structure optimization consists of fine-tuning the so-initialized parameters (see Fig. 5) using a Levenberg–Marquard version of the gradient back-propagation combined with a least square estimate (LSE). The cost function is the error between network outputs and the desired outputs relating the PKM dynamic behavior

$$E = \frac{1}{2}\sum_{k=1}^{K} \|\mathbf{o}^k - \boldsymbol{\pi}^k\|^2 = \frac{1}{2}\sum_{k=1}^{K}\sum_{j=1}^{J}\sum_{l=1}^{4} \left[\mathbf{o}_{jl}^k(a_l, b_l, \rho_l, \sigma_l) - \boldsymbol{\pi}_{jl}^k\right]^2 \quad (30)$$

where $\mathbf{o}^k$ is the computed output from the fuzzy system $\boldsymbol{\pi}_d^k$ the k^{th} desired output associated with the k^{th} training and dataset entry and J the number of rules.

IV. Simulation Results

A. Training Protocol

In order for the neuro-fuzzy system to learn mostly of the robot workspace, the training protocol is achieved as follows.

1) Because the order in which the points are presented to the network affects the speed and quality of convergence and since the robot interpolates between sampling points to build the entire path from a start to end points, the trajectory generation is achieved while ensuring singularity and torque limits avoidance and satisfaction of other constraints.
2) Two hundred ALD trajectories for different initial and final EE Cartesian positions are generated. Each trajectory is sampled in 20 discretization points, leading to 4000 data points to perform NeFuMOP training. 80% are used for training, 10% for testing, and 10% for checking.

B. Offline Planning and Dataset Generation

For offline planning, the focus is on time-energy constrained trajectory planning with the following objectives.

1) Minimize traveling time and kinetic and electric energy while avoiding singularity during the motion.
2) Satisfy several constraints related to limits on joint positions, rates, accelerations and torques. More kinematic related analysis for a similar case study may be found in [7], [14]. Viscous and dry frictions are considered. The function $arctg$ is used to approximate the sign function. The program is coded in Matlab. A unity value is used for each weight factor in the cost function. The following numeric values are used: The EE mass is $m_{EE} = 200.0$ kg, that of each leg is $m_l = 570.5$ kg, and that of the slider is $m_s = 70$ kg.

The simulated platform is described in details in [7], [14]. The platform radius is $r = 0.75$ m the distance $R = 1.2030$ m and the strut length $L = 1.9725$ m. Table I shows the limits of the robot workspace, actuator torques and sampling periods. As for the ALD parameters, the following values had been taken $\omega_s = 0.5$, $\eta_s = 0.5$, $\alpha_w = \alpha_\eta = 0.4$, $\beta_w = \beta_\eta = 0.4$, $w_0 = \eta_0 = \eta_{10} = 10^{-2}$, $\gamma_1 = 0.25$ $w^* = \eta^* = \eta_1^* = 10^{-5}$, $\gamma_2 = 1.2$, $\upsilon = 0.01$, $\bar{\upsilon} = 0.3$. The initial Lagrange multipliers $\boldsymbol{\sigma}_0, \boldsymbol{\rho}_0$ are set to zero. A sample trajectory starts from an initial Cartesian state position $x_0 = -0.7$, $y_0 = -0.1$ to

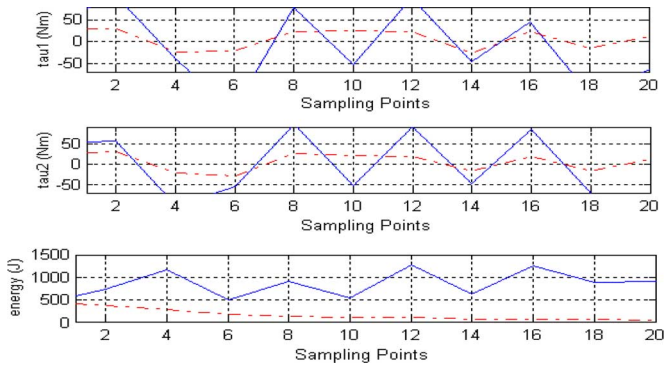

Fig. 7. Initial and ALD torques and energy.

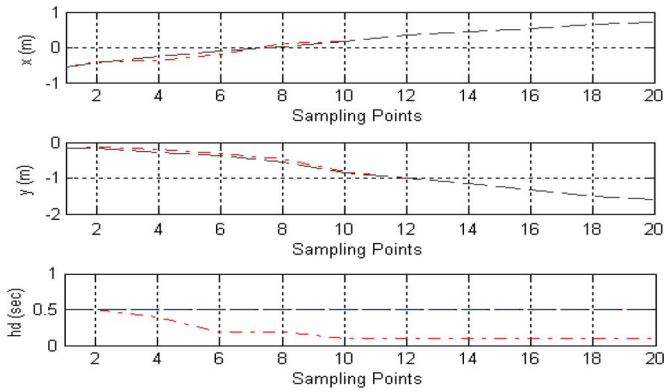

Fig. 8. Fixed versus optimized sampling periods.

a final position $x_T = 0.7$, $y_T = -1.6$ (in meters). The initial and final linear and angular velocities are set to zero. The maximum velocity is 0.2 m/s and maximum acceleration is 2 m/s^2. The maximum allocated time for this trajectory is 10 s. Fig. 6 shows simulation results for both initial kinematic and augmented Lagrangian solutions. The first plot from the top shows the displacement along x-axis of the EE point of operation. The second plot shows the displacement along y-axis of the EE point of operation. The third shows the instantaneous values of consumed time to achieve the trajectory. In Fig. 7, the first and second plots from the top show the instantaneous variations of joint torques, while the third one shows the instantaneous values of the consumed energy. It is noteworthy that although the initial solution is kinematically feasible, when the corresponding torques is computed considering the dynamic model and forces one obtains torque values that quickly get outside the admissible domain resulting in high values for energy cost. With the augmented Lagrangian however, with four inner and seven outer iterations, the variations of the energy consumption increase smoothly and monotonously. Fig. 8 displays the simulation outcomes for only the energy criterion (i.e., the time weight is set to zero, so the sampling period is kept constant), one gets a 21% faster trajectory with time-energy criterion compared to a trajectory computed with only the minimum-energy criterion (dotted line). The multiobjective trajectories are smoother than minimum time and faster than only minimum energy trajectories. This allows very good reference trajectories for online

TABLE II
CONVERGENCE HISTORY OF MINIMUM TIME-ENERGY PLANNING

NDisc	NPrimal	NDual	CPU(sec)	t_T (sec)	Energy(J)	APEq	APIneq
10	4	7	101.62	8.41	7981.54	2.510^{-3}	1.210^{-3}
20	4	7	111.25	7.11	4906.58	10^{-3}	2.10^{-3}
20	5	10	121.54	3.32	3548.88	3.910^{-4}	3.410^{-4}
30	5	10	143.83	1.85	3921.41	3.10^{-5}	2.10^{-5}

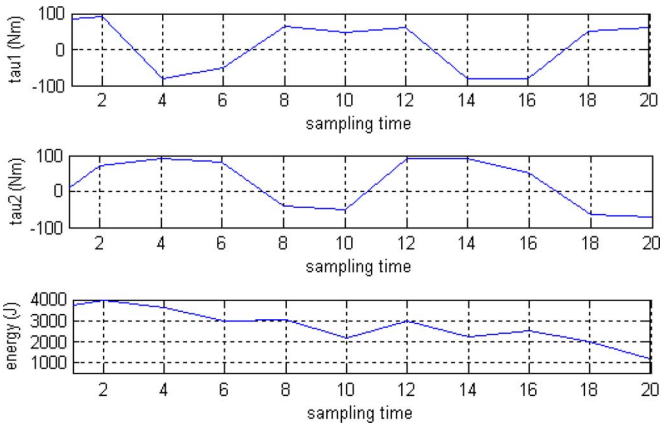

Fig. 9. ALD torques and energy with imposed passages.

planning. To analyze with respect to ALD parameters, Table II shows comparison of results for different simulation parameters of ALD, where NDisc stands for the number of discretisations, NPrimal is the number of inner optimization loops, NDual is the number of outer optimization loops, $t_T = \sum_{k=1}^{N} h_k$ is the total traveling time, Energy $= \sum_{k=1}^{N} [(\boldsymbol{\tau}_k \boldsymbol{U} \boldsymbol{\tau}_k^T + \boldsymbol{x}_{2k} \boldsymbol{Q} \boldsymbol{x}_{2k}^T)]$ is the consumed electric and kinetic energy, and APEq and APIneq for achieved precision for equality and inequality constraints, respectively. The values shown for the total traveling time t_T, Energy, and AP correspond to those computed for the last outer iteration. As for imposed passages through prespecified poses, the same scenario is simulated, while constraining the EE to pass through the following positions: (0.0, -1.4), (0.4, -1.1), (0.5, -1.0), all in meters. Fig. 9 shows the torques corresponding to ALD trajectory with imposed passages. With seven primal iterations and nine dual iterations we obtained a precision of 7.10^{-4}, which confirms the well known performance of ALD in constraints satisfaction. A detailed study on the impact of each criterion on PKM performance, as provided in [7] shows that the multiobjective trajectory is smoother than minimum time and faster than minimum energy trajectories, making it very suitable for online planning. Another important issue is sensitivity analysis [23], as PKMs are strongly nonlinear and coupled mechanical systems, several of these parameters such as inertial parameters are known only approximately or may change. To assess how robust the proposed approach to the parameter changes is a first test is performed with a modified value of the EE mass. Fig. 10 shows the ALD simulation by varying the EE mass to $m_{EE} = 250$ kg. Several scenarios had been implemented and 200 multiobjective trajectories with different starting and ending points are generated within the admissible domain.

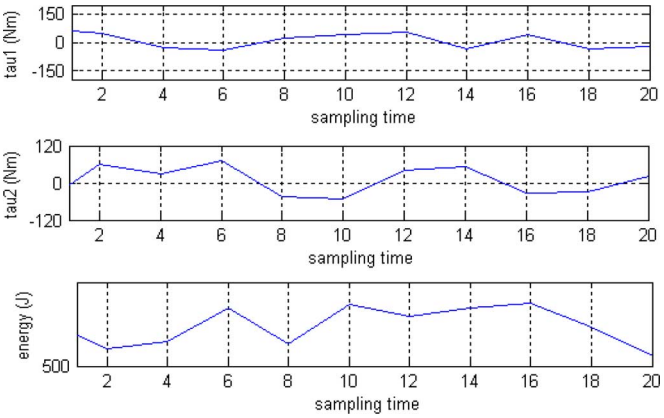

Fig. 10. ALD torques and energy for varied EE mass.

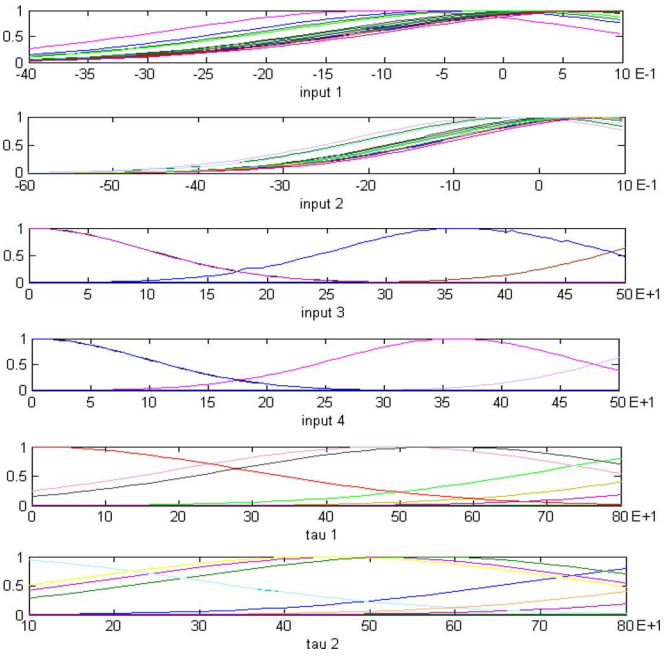

Fig. 11. Optimized membership functions for x, $\dot{x}$, y, $\dot{y}$, $\boldsymbol{\tau}_1$, $\boldsymbol{\tau}_2$.

C. NeFuMOP Performance

A subtractive clustering algorithm partitions the so-generated input/output dataset [22]. The subtractive clustering parameters had been set after several trials to the following: The upper acceptance threshold for a data point to be a cluster center is 1.0. The lower rejection ratio is 0.7. The cluster radius is 0.9, and the squash parameter is 0.5. Fig. 11 shows the optimized membership functions of the inputs $(x_i, y_i, \dot{x}_i, \dot{y}_i)$ and outputs noted $\boldsymbol{\tau}_1$ and $\boldsymbol{\tau}_2$. The learning results show very good performance with an error for $\boldsymbol{\tau}_1$ of order 0,0381 and 0.029 for $\boldsymbol{\tau}_2$ in 40 training epochs as illustrated in Figs. 12 and Table III. NeFuMOP required 283 seconds to learn from the overall 4000 data points and capture the PKM dynamic behavior. To test the learning, and generalization abilities of NeFuMOP, a preliminary test is performed with small modifications (5%) of each of the 10% testing data points. Table IV shows the learning performance achieved by NeFuMOP with the modified data reaching an error of order 10^{-2} after 40 training epochs, which

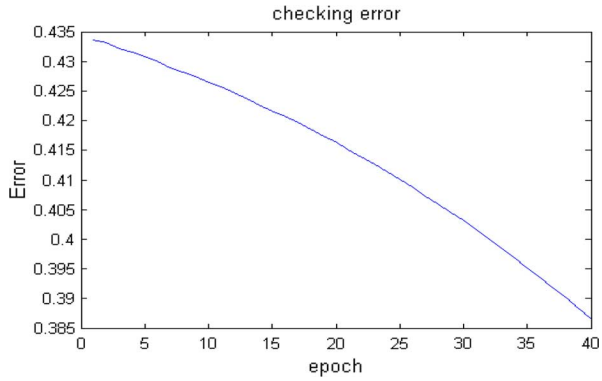

Fig. 12. NeFuMOP learning performance for $\boldsymbol{\tau}_1$.

TABLE III
PERFORMANCE OF NEFUMOP

4000 Data entries	RMS Error (after 50 epochs)
Training data (90%)	0.015441
Training data (10%)	0.0018531

TABLE IV
PERFORMANCE OF NEFUMOP (WITH DISTURBED TESTING DATA)

4000 Data entries	RMS Error (after 60 epochs)
Training data (90%)	0.01517
Testing data (10%)	0.002391
Checking data (10%)	0.002612

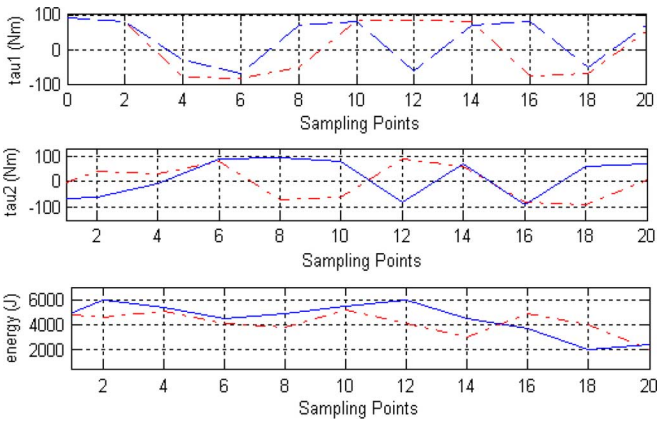

Fig. 13. Test trajectory, NeFuMOP, and ALD.

highlights very good performance. Another test is performed to assess the generalization capabilities of the proposed system. It is made on a trajectory that had been learned using the ALD offline technique, which consisted of moving the PKM-EE along a circle, with a radius (0, −1.4) (m). The ALD technique was used to offline plan this trajectory. The actuator torques and sampling period variations are obtained from the outcomes $\mathbf{o}^k$ of NeFuMOP. The trajectory and consumed energy are shown in Fig. 13. It shows very good generalization capabilities of NeFuMOP. At this stage, we point out that because NeFuMOP interpolates within the provided trajectories in training dataset, the optimality generated with NeFuMOP. Nonetheless, this constitutes a great advantage as compared to as to date

approaches using neuro-fuzzy techniques, which provide trajectories based on pure feasible if/then rules, without optimality or near optimality with respect to given criteria nor constraints satisfaction. This approach had been implemented on serial manipulators and mobile robots and it gives very satisfactory results. In an ongoing work, it is being implemented on a 6-DOF Stewart–Gough platform.

V. Conclusion and Discussion

The basic contribution of this brief is a systematic design of a multiobjective trajectory planning system for parallel kinematic machines. This system is built upon the outcomes of an offline planning optimizing traveling time and electric and kinetic energy and based on PKMs kinematics and dynamics as well as velocity, accelerations, actuator torques and workspace limits while avoiding singularities. The augmented Lagrangian algorithm is implemented on a decoupled dynamics of the PKM. According to simulation results, the offline planning technique produced very good results with singularity-free smoother trajectories compared to minimum time, kinematic-based control techniques or other optimization techniques like penalty methods. This makes it very suitable for training data generation to use to achieve online motion planning. The neuro-fuzzy model built upon the previous offline generated dataset to achieve online multi-objective motion planning had shown very satisfactory results as well. The high problem conditioning abilities of ALD associated with the modeling and learning capabilities of neuro-fuzzy networks had been proved effective to cope with difficult nonlinear dynamics and kinematics of the system, making it possible for the development of an online multiobjective motion planning for parallel robots.

Appendix A

The condition number κ is defined as $1 \leq \kappa = (\sigma_1/\sigma_2) \leq \infty$, where σ_1 and σ_2 are the minimum and maximum singular values of the Jacobian associated with a given posture.

For the parallel robot under study, one has $\kappa = \sqrt{A + \sqrt{B}/A - \sqrt{B}}$, with

$$A = p^2 + q^2 + 2,\ B = (p^2 + q^2)^2 + 8pq + 4$$

and

$$p = \frac{x - r + R}{y - y_1}, \quad q = \frac{x + r - R}{y - y_2}.$$

Appendix B

With the defined control law by (18) and from (6), one has (after removing function arguments):

$$\boldsymbol{M}\dot{\boldsymbol{x}}_2 + \boldsymbol{N}\boldsymbol{x}_2 + \boldsymbol{G} + \boldsymbol{J}^T\boldsymbol{f}_c = \boldsymbol{M}\boldsymbol{J}^{-1}\boldsymbol{v} + \boldsymbol{N}\boldsymbol{x}_2 + \boldsymbol{G} + [\boldsymbol{J}^T - \boldsymbol{M}\boldsymbol{J}^{-1}\boldsymbol{C}]\boldsymbol{f}_c - \boldsymbol{M}\boldsymbol{J}^{-1}\dot{\boldsymbol{J}}\boldsymbol{x}_2$$

$\boldsymbol{C}$ being an 2×2 diagonal matrix, and since $\boldsymbol{M}(\boldsymbol{x})$ is invertible

$$\dot{\boldsymbol{x}}_2 = \boldsymbol{J}^{-1}\boldsymbol{v} - \boldsymbol{J}^{-1}[\boldsymbol{C}\boldsymbol{f}_c + \dot{\boldsymbol{J}}\boldsymbol{x}_2].$$

This gives

$$\boldsymbol{J}\dot{\boldsymbol{x}}_2 + \dot{\boldsymbol{J}}\boldsymbol{x}_2 = \boldsymbol{v} - \boldsymbol{C}\boldsymbol{f}_c$$

which yields, using the forward kinematics to

$$\ddot{\mathbf{x}} = \boldsymbol{v} - \boldsymbol{C}\boldsymbol{f}_c.$$

Acknowledgment

The author would like to thank the anonymous reviewers for their valuable comments and King Fahd University of Petroleum and Minerals for general support.

References

[1] V. E. Gough, "Contribution to discussion of papers on research in automobile stability, control and type performance," in *Proc. Auto Div. Inst. Mech. Eng. Pt. D, J. Automob. Eng.*, 1956, pp. 392–395.
[2] J. P. Merlet, *Parallel Robots*. Dordrecht, the Netherlands: Kluwer, 2000.
[3] K. Harib and K. Srinivasan, "Kinematic and dynamic analysis of Stewart platform-based machine tool structures," *Robotica*, vol. 21, no. 5, pp. 541–551, 2003.
[4] I. Pietsch, M. Krefft, O. Becker, C. Bier, and J. Hesselbach, "How to reach the dynamic limits of parallel robots? An autonomous control approach," *IEEE Trans. Autom. Sci. Eng.*, vol. 2, no. 4, pp. 369–380, 2005.
[5] H. Abdellatif and B. Heimann, "A novel multiple-heuristic approach for singularity-free motion planning of spatial parallel manipulators," *Robotica*, vol. 26, no. 05, pp. 679–689, 2008.
[6] M. Hay and J. A. Snyman, "Methodologies for the optimal design of parallel manipulators," *Int. J. for Num. Methods Eng.*, vol. 59, no. 1, pp. 131–152, 2004.
[7] A. Khoukhi, L. Baron, and M. Balazinski, "Constrained multi-objective trajectory planning of parallel kinematic machines," *Robot. Comput. Integr. Manuf.*, vol. 25, pp. 756–769, 2008.
[8] S. J. Ho, S. Y. Ho, M. H. Hung, L. S. Shu, and H. L. Huang, "Designing structure-specified mixed $\mathrm{H}_2/\mathrm{H}_\infty$ optimal controllers using an intelligent genetic algorithm IGA," *IEEE Trans. Control Syst. Technol.*, vol. 13, no. 6, pp. 1119–1124, 2005.
[9] Y. L. Sun and M. J. Er, "Hybrid fuzzy control of robotics systems," *IEEE Trans. Fuzzy Syst.*, vol. 12, no. 6, pp. 755–765, 2004.
[10] A. Khoukhi and A. Ghoul, "On the maximum dynamic stress search problem for robot manipulators," *Robotica*, vol. 22, no. 5, pp. 513–522, 2004.
[11] A. Khoukhi and K. Demirli, "A data-driven minimum energy fuzzy parking of car-like mobile robots," in *Recent Advances in Control Systems, Robotics, and Automation*, 3rd ed. Italy: International SAR, 2009, pp. 142–150.
[12] H. T. Mok, C. W. Chan, and W. K. Yeung, "Neuro-fuzzy network-based adaptive nonlinear PI controllers," *Control Intell. Syst.*, vol. 34, no. 3, pp. 1557–1568, 2006.
[13] M. Al-Khatib and J. J. Saade, "An efficient data-driven fuzzy approach to the motion planning problem of a mobile robot," *Fuzzy Sets Syst.*, vol. 134, no. 1, pp. 65–82, 2003, 16.
[14] X. J. Liu, Q. M. Wang, and J. Wang, "Kinematics, dynamics and dimensional synthesis of a novel 2-DOF translational manipulator," *J. Intell. Robotic Syst.*, vol. 41, pp. 205–224, 2004.
[15] Hubert and J.-P. Merlet, "Static of parallel manipulators and closeness to singularity," *ASME J. M. Robot.*, vol. 1, p. 1, 2009.
[16] P. L. Lions, *Generalized Solutions of Hamilton-Jacobi Equations*. Boston, MA: Pittman, 1982.
[17] D. P. Bertsekas, *Non-Linear Programming*. Nashua, NH: Athena Scientific, 1995.
[18] A. Isidori, *Non-Linear Control Systems*, 3rd ed. London, U.K.: Springer, 1995.
[19] L. V. Fausett, *Applied Numerical Analysis Using MATLAB*, 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, 2008.
[20] A. Khoukhi, L. Baron, M. Balazinski, and K. Demirli, "Hierarchical neuro-fuzzy optimal time trajectory planning for redundant manipulators," *Eng. App. Art Intell.*, vol. 21, no. 7, pp. 974–984, 2008.
[21] J. S. R. Jang, C. T. Sun, and E. Mizutani, *Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence*. Englewood Cliffs, NJ: Prentice-Hall, 1996.
[22] S. L. Chiu, "Fuzzy model identification based on cluster estimation," *J. Integr. Fuzzy Syst.*, vol. 2, pp. 267–278, 1994.
[23] *Sensitivity Analysis*, A. Saltelli, K. Chan, and E. M. Scott, Eds. New York: Wiley, 2000.