Multi-Agent System Inspired Distributed Control of a Serial-Link Robot

Submitted: 20th January 2019; accepted: 20th November 2019

S Soumya, K R Guruprasad

DOI: 10.14313/JAMRIS/1-2020/4

Abstract: *Inspired by the multi-agent systems, we propose a model-based distributed control architecture for robotic manipulators. Here, each of the joints of the manipulator is controlled using a joint level controller and these controllers account for the dynamic coupling between the links by interacting among themselves. Apart from the reduced computational time due to distributed computation of the control law at the joint levels, the knowledge of dynamics is fully utilized in the proposed control scheme, unlike the decentralized control schemes proposed in the literature. While the proposed distributed control architecture is useful for a general serial-link manipulator, in this paper, we focus on planar manipulators with revolute joints. We provide a simple model-based distributed control scheme as an illustration of the proposed distributed model-based control architecture. Based on this scheme, distributed model-based controller has been designed for a planar 3R manipulator and simulations results are presented to demonstrate that the manipulator successfully tracks the desired trajectory.*

Keywords: *Model-based control, distributed control, manipulator control*

1. Introduction

Moving the end-effector along the desired trajectory is one of the fundamental problems in robotics. Designing a controller guaranteeing desired performance for this manipulator motion control problem is a challenging task owing to highly nonlinear and coupled nature of its dynamics.

Nonlinear model-based controllers [5, 28] use the concept of feedback linearization. In these control schemes, the trajectory tracking performance level is uniform across the state space. However, one of the major limitations of these control schemes is the fact that they require online computation of the dynamic equations. Being a coupled multi-input multi-output (MIMO) and nonlinear system, manipulator dynamic equations are computationally intense, particularly with higher degrees-of-freedom. A feed-forward scheme known as computed torque control approach [24], where the dynamic equations are pre-computed along the desired trajectory, may be used to reduce the computational lead time. The error dynamics is close to that with the nonlinear model-based controller when the tracking error is small. However, with higher tracking error, the performance starts degrading, as the nonlinear terms do not cancel out. In addition to the increased spatial (memory) complexity, such a scheme cannot be used in situations where the trajectory is generated online. In some situations, the dynamic model may not be available fully or approximated even if known to reduce computation. Several approaches such as robust control [27], adaptive control [22], model predictive control [19], Artificial Neural Networks [13,15], Fuzzy logic controllers, or a combination known as Adaptive Network-based Fuzzy Inference System (ANFIS) [6], etc., have been used in such scenarios.

As most above controllers are typically computationally expensive, several independent-joint controllers have been proposed in the literature, where a dedicated controller is used to control the motion of each of the joints. These control schemes are also referred to as decentralized control and sometimes wrongly as distributed control schemes. Independent joint PD/PID control [5] is the simplest decentralized control scheme. Seraji [21] proposed a decentralized control scheme without using a manipulator dynamic model. Each joint controller, along with PID control law, uses a feed-forward loop with adaptive gains. In [10] the authors propose a decentralized linear control using the control input computed in the previous time instance to estimate the coupling terms in the manipulator dynamics. Their control law approaches the model-based control law as the time delay (sampling time) approaches zero. An adaptive version of this control law is presented in [3]. A nonlinear adaptive decentralized controller is proposed in [16], where the author attempts to account for nonlinear coupling by using decentralized cubic feedback. In [17] the author uses a robust nonlinear feedback term in addition to the decentralized cubic feedback to a decentralized PD control law. In [11] an adaptive decentralized controller using adaptive variable structure compensations has been proposed. A decentralized robust control scheme is proposed in [26]. Here, the authors consider the unmodelled coupled dynamics as disturbances and use a disturbance observer (DOB) to compensate for the same.

In this work, we address the problem of control of manipulator when its dynamics is known completely. As we have seen, the nonlinear model-based controller or similar techniques are best suited in terms of provable performance guarantee. However, when we consider a high degree of freedom manipulator, such as a hyper-redundant planar manipulator, say, the computational cost with the model-based controller substantially increases. Though the decentralized control scheme may result in lower computational cost, in these schemes the effect of dynamic coupling between links cannot be accounted for, which in turn leads to performance degradation.

In this paper, inspired by distributed multi-agent systems, we propose a distributed control architecture and present a simple distributed control law based on the proposed architecture for a serial-link robot with revolute joints. Some of the basic concepts have been reported in [23]. This paper provides a more detailed presentation along with simulation results demonstrating the proposed distributed control scheme.

2. Multi-Agent Systems and Distributed Control

Multi-agent systems (MAS) such as multi-robotic systems (MRS), where multiple cooperating simple agents such as mobile robots, are increasingly being used to solve many complex problems, such as search and rescue [7], landmine detection [8, 4], etc. In the context of a multi-agent system, centralized, decentralized, and distributed control architectures have been used. Figure 1 illustrates these three architectures.

In the case of the centralized control architecture, a single central controller controls all the agents. This architecture suffers from high computational load and communication overhead. Further, failure of the controller leads to failure of the entire system. In the case of the decentralized architecture, each agent is controlled by an individual controller. There is no interaction between the controllers, though the agents may interact with each other. The major advantage of this architecture over the centralized architecture is reduced computational load as the multiple agent level controllers share the load. However, as the individual controllers do not communicate among themselves, the coupling between the agents is not considered. In a distributed architecture, the agent level controllers cooperate by communicating among themselves, thus taking care of the coupling/interactions among the agents. The distributed architecture results in a reduced computational overhead without compromising on the performance. Further, the distributed architecture does not suffer from the single point failure as in the case of the centralized architecture and is typically robust to the failure of a few individual controllers, provided that the multi-agent system itself is robust to failure of a few individual agents. Though there are important subtle differences between the decentralized and distributed architectures as discussed here, these two terms have been used interchangeably in the literature.

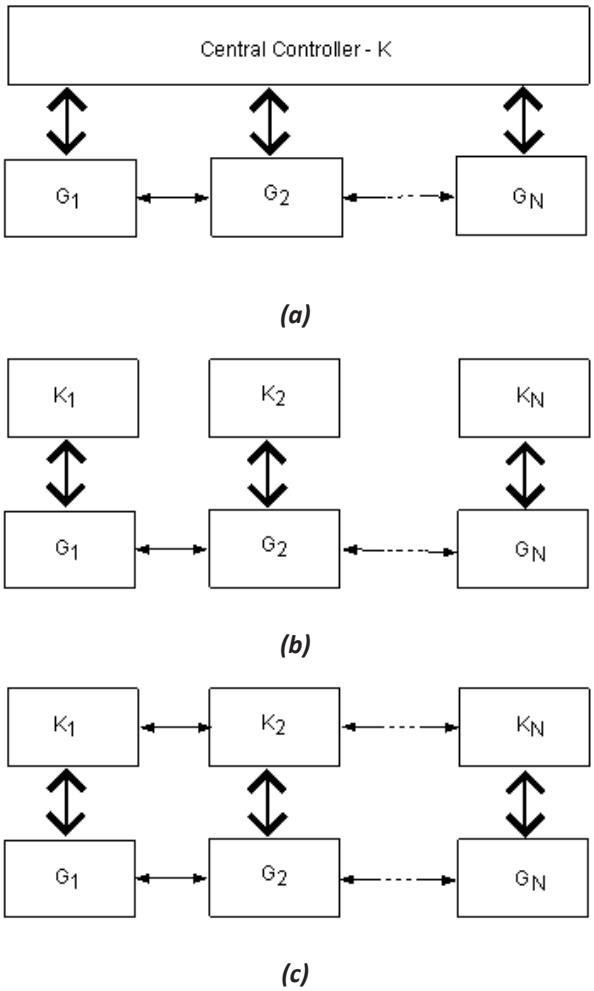


Fig. 1. (a) Centralized (b) decentralized, and (c) distributed control architecture used in multi-agent or networked systems

Remark 1. The fundamental difference between the centralized control architecture and the distributed/decentralized control architectures is that a single controller is used in the former (centralized architecture) and a dedicated controller for agent/subsystem is used in the latter (decentralized and distributed architectures). The fundamental difference between a decentralized and a distributed architecture is that the former does not require communication between the individual (agent level) controllers while the latter allows/requires such communication. Further, though only local communication is used in a typical distributed control architecture, a control law requiring complete communication (that is, with all the other agent-level controllers, not necessarily restricted to neighboring controllers) may also be implemented in a distributed architecture as long as the communication graph is connected, using a multi-hop distributed communication.

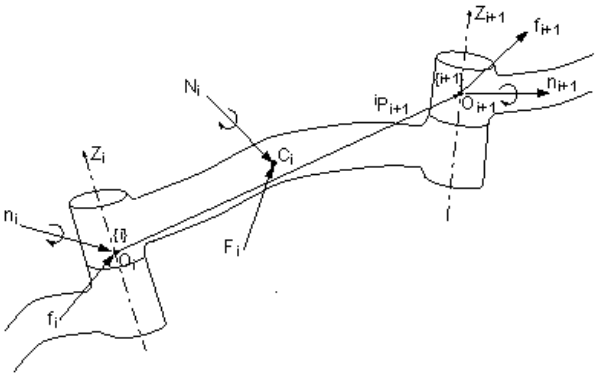


Fig. 2. Connected links exert forces and moments through the joints

Manipulator as a multi-agent system. A serial-link manipulator consists of several links with joints that allow motion between them [5]. Apart from allowing motion, links also physically interact in terms of interactive forces and moments between them through these joints. As illustrated in Fig. 2, the link (*i-1*) exerts a force f_i and a moment n_i on the link *i*. Similarly, the link (*i+1*) exerts a force $-f_{i+1}$ and a moment $-n_{i+1}$ on the link i. With these interactive forces and moments from connected links, the i^{th} link experiences a net force of F_i and a net moment of N_i. The actuator applies a moment (or force in the case of prismatic joints) about (along) the joint axes Z_i. We may consider a 'joint-link pair' as a subsystem or an agent, interacting with other subsystems/agents. In this sense, a serial-link robot is a multi-agent system. However, unlike in a typical multi-agent system, where the coupling between any two agents is at the behavioral level, interactions between the agents (joint-link pairs) in a manipulator are at the physical level. Note that the i^{th} link directly interacts only with the neighboring links *i-1* (through the joint i) and *i+1* (through the joint *i+1*). However, interaction of link *i+2* with *i+1* is experienced on the link *i* through the link *i+1*. In this way, motion of (and the force/torque on) every link affects every other link. Such an indirect interaction is also seen in distributed multi-agent systems. Direct local interactions lead to interaction between every (connected) agent.

3. Computational Cost of Manipulator Dynamics

The equation modeling the dynamics of a serial link manipulator has the form [5]:

$$\tau = M(\theta)\ddot{\theta} + C(\theta,\dot{\theta}) + G(\theta) \qquad (1)$$

Here, τ is the vector of joint torques with size *N X 1*; $M(.)$ is the mass matrix with size *N* X *N*; θ; $\dot{\theta}$; and $\ddot{\theta}$ are joint angle, velocity, and acceleration vectors, respectively, all of size *N* X *1*; $C(.\ ,\ .)$ is the vector involving centripetal and Coriolis accelerations, of size *N* X *1*; and $G(.)$ is the vector of gravity terms of size *N* X *1*. Here, *N* is the degrees-of-freedom of the manipulator. The standard model-based control law is [5]:

$$\tau = M(\theta)\left(\ddot{\theta}_d + K_V\dot{E} + K_P E\right) + C\left(\theta,\dot{\theta}\right) + G(\theta) \qquad (2)$$

Here, θ_d is the vector of desired joint angles, $E = \theta_d - \theta$ is the tracking error, and K_p and K_v are diagonal matrices of controller gains.

The model-based nonlinear control law given by Eqn. (2) uses the dynamic model of the manipulator for computing the control input. Thus, the computational cost of the dynamic equations dictates the frequency at which the control input can be updated. Higher the computational cost, the higher is the computational lead-time. The computational cost associated with the dynamic equations of a manipulator increases with degrees-of-freedom.

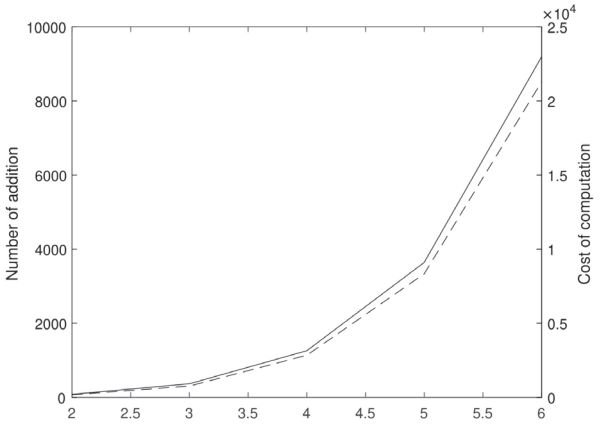


Fig. 3. Variation of number of arithmetic operations (shown with dashed line) and total cost of computation (shown with solid line) of the dynamic equations of a planar manipulator with the degrees-of freedom

We carried out a simple analysis to find out how the number of computations and hence the computational cost depends on the degrees-of-freedom, using Maple. We considered planar manipulators with degrees-of-freedom from 2 to 6. We used iterative Newton- Euler formulation method to obtain the manipulator dynamics using Maple. In computing computational cost, we have considered cost of addition/subtraction as 1 unit. Multiplication operation is definitely computationally more expensive than addition, though the actual relative cost depends on the algorithm used and the processor itself. Here, for the purpose of comparative cost analysis, we have assumed that the computation of multiplication operation is four times as expensive as that of addition. As trigonometric terms appear at most twice per degree of freedom, in the form of cosine and sine, we have not considered them, though they are computationally more expensive. The number of arithmetic operations and the corresponding cost involved in computation of the dynamic equation of a planar manipulator with revolute joints for different degrees-of-freedom are plotted in Fig. 3. Based on these results, we

may obtain an empirical relationship for number of arithmetic operations (N_{Arith}) in the dynamic equation of a planar manipulator as a function of the degrees-of-freedom N as:

$$N_{Arith} = 35.583N^4 + 370.5N^3 + 1676.9N^2 + 3424N + 2605 \quad (3)$$

which is polynomial in N.

A similar trend is expected in a general serial manipulator, or even parallel and hybrid manipulators, though the dynamic coupling (between neighboring links) effect is the maximum in the case of a planar serial manipulator. The computational issues may not be very crucial for control of manipulators with small degrees-of-freedom or those which can use high performance processors for implementation of the control law. However, as the degrees-of-freedom increases, particularly in redundant or hyper redundant manipulators, higher computational effort may start affecting the trajectory tracking performance.

4. Distributed Manipulator Control Architecture

Now we propose a distributed control architecture as illustrated in Fig. 4 for a manipulator, exploiting the distributed nature of manipulator dynamics as discussed earlier. Here, each joint-link agent is controlled by a dedicated joint level controller. While the joint-link agents interact directly with neighboring agents and indirectly with other agents, the joint level controllers interact with the neighboring controllers directly in the form of communication and indirectly with all other controllers. A class of torque feedback based manipulator control schemes [1, 9, 14, 20] has distributed architecture discussed here. However, as these schemes require torque sensors to measure the motor torques at each joint, these are not suitable for a large number of existing manipulators which lack such a sensing capability.

There has been some attempt in the literature to perceive manipulator as a multi agent system. In [18] the authors consider a joint-link pair as an agent and use the multi-agent system concept for manipulator control. These agents are software agents rather than physical agents. Further, this paper addresses kinematics rather than dynamic control of the manipulator. The inverse kinematics problem is solved using a distributed architecture that provides input to a high-level controller. Bohner et al [2] present a reactive planning and control system for redundant manipulators. Here a 'joint-agent' is responsible for planning and controlling the motion of one joint, by integrating sensor data, such as tactile sensors. Jia et al [12] proposed distributed architecture for a light space manipulator. However, they do not consider manipulator dynamics. Tsuji et al [25] presented a distributed control for redundant control for redundant manipulators based on a concept of virtual arms. Though the authors present control at dynamic level, the subsystems here are virtual arms rather than the join-link pairs.

Now we provide a simple model-based distributed control scheme based on the proposed distributed architecture, without use of any additional sensors.

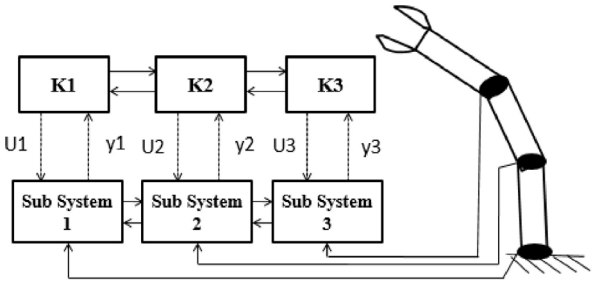


Fig. 4. Distributed manipulator control architecture

A simple distributed control scheme. The model-based control law given in Eqn. (2) represents a control law using the centralized control architecture, where a single central controller computes control inputs, τ_i, $i=1,...,N$, for all the joints. Note that here all the variables τ, θ, $\dot{\theta}$, $\ddot{\theta}$, E, $\dot{E}$, etc. are obtained in time t. Consider a simple distributed control law [17] for the i^{th} joint level controller as:

$$\tau_i = \sum_{J=1}^{N} M_{ij}\ (\theta)\left(\ddot{\theta}_{dj} + K_{Vj}\ddot{E}_j + K_{Pj}E_j\right) + C_i\left(\theta,\dot{\theta}\right) + G_i\left(\theta\right) \quad (4)$$

Here, the index j (or i) is used to indicate the corresponding component of a vector, and M_{ij} is the j^{th} element in i^{th} row of M. Note that the Eqn. (4) is the i^{th} component of the model-based control law as given in Eqn. (2). Let K_i be the i^{th} joint level controller using the control law given in the Eqn. (4). For the model-based control law given in here, we make following observations:

1. The output of each controller K_i is τ_i, the control input to the i^{th} joint.
2. Controller K_i requires inputs from other joint-link agents which it may receive through the corresponding joint level controllers K_j, $j \neq i$.
3. The controller K_i is connected to neighboring controllers K_{i-1} and K_{i+1}, in the sense that it can send and receive signals.
4. Now the adjacency graph of K_i is connected.
5. Thus, the controller K_i can receive (send) signals from (to) any controller K_j, $j \neq i$ through a distributed (multi-hop, if required) communication.
6. The adjacency graph formed by the joint level controllers is identical to that formed by the joint-link agents.

Thus, the control architecture, where each joint level controller K_i controls the i^{th} joint while obtaining necessary information (such as feedback values of joint states and the desired states), from the neighboring joint level controllers has a natural distributed architecture, which in fact is the result of the distributed nature of the manipulator dynamics. As observed in Remark 1, the joint level controllers are allowed to

communicate directly with the immediate neighbors, and as every joint level controller is indirectly connected to every other controller, the required information may be obtained through (multi-hop) distributed communication between the joint level controllers. Hence, though the control law corresponding to K_i may contain terms corresponding to every joint/link, not only those corresponding to the immediate neighboring agents, the control scheme based on the Eqn. (4) is naturally amenable for a distributed implementation.

Theorem 1. The control law given by the Eqn. (4), with positive gains, makes the links of the manipulator whose dynamics is given by the Eqn. (1), follow the desired trajectory $\theta_d(t)$, asymptotically.

Proof. The closed-loop error dynamics may be obtained from Eqn (4) and Eqn. (1) as,

$$\ddot{E}_j + K_{Vj}\ddot{E}_j + K_{Pj} \qquad \forall J \in 1,2,\ldots,N \tag{5}$$

Thus, we have $E_j \to 0$, as $t \to \infty$, $\forall j \in \{1,.....,N\}$, for positive gains.

Remark 2. Here, the closed-loop error dynamics is identical to that obtained using the conventional model-based control scheme given by Eqn. (2). This is not surprising for two reasons. First, the joint level controllers K_i and the distributed control scheme presented here are based on the control laws given in Eqn. (4), which itself is based the control law given in Eqn. (2). Second, any control law that cancels the nonlinear and coupled dynamics using feedback, or in other words, achieves feedback linearization, should result in linear decoupled closed-loop error dynamics as given in Eqn. (5). The contributions here are: identifying a natural distributed nature of the model-based control law given in Eqn. (2), presenting a control scheme that is amenable for implementation in the distributed control architecture, and obtaining feedback linearization leading to guaranteed state independent trajectory tracking performance, unlike the decentralized (or independent joint) control schemes presented in the literature.

Remark 3. The distributed control scheme given by the Eqn. (4) is only a simple example of a control scheme/law that can be implemented in the proposed distributed manipulator control architecture. In principle, several model-based control schemes may be designed within the proposed distributed architecture.

Distribution effectiveness. The main objective of the distributed control law for a manipulator being reduction in the cost of computation involved in the control law, we define a quantity known as distribution effectiveness. Let C_{Ti} be the computational cost associated with the i^{th} joint level controller, and C_{Tc} be that associated with the corresponding centralized controller. Now we define the distribution effectiveness for N degree of freedom robot as

$$\eta_d = \frac{C_{TC}/N}{\max_i\left(C_{Ti}\right)} \tag{6}$$

In an ideal situation, when the computation is distributed uniformly among the individual controllers, we get $\eta_d = 1$.

5. Discrete-Time Implementation: Effect of Time

The model-based (centralized) control law as given in Eqn. (2) is in continuous time domain. However, in reality, this control law is realized in discrete time. The model-based control law in discrete time is given by

$$\begin{aligned}\tau(t) = M\left(\theta\left(t-T_d\right)\right)\left(\ddot{\theta}_d\left(t-T_d\right) + K_V\dot{E}\left(t-T_d\right)\right.\\ + K_P E\left(t-T_d\right) + C\left(\theta\left(t-T_d\right),\dot{\theta}\left(t-T_d\right)\right)\\ + G\left(\theta\left(t-T_d\right)\right)\end{aligned} \tag{7}$$

Where, T_d is the time delay introduced due to the sampling time T_d. The sampling time depends on the time required to compute the control law Eqn. (7), along with any other processing required. Note that with the discrete control law given in Eqn. (7), feedback linearization is not achieved. We can achieve the feedback linearization only when $T_d = 0$. However, due to the continuity of the dynamics of the manipulator and the model-based control law, tracking performance is expected to degrade gracefully with increasing T_d. Now consider the discrete-time distributed control law based on that given in Eqn. (4),

$$\begin{aligned}\tau_i(t) = \sum_{J=1}^{N} M_{ij}\ \left(\theta\left(t-T_d^d\right)\right)\left(\ddot{\theta}_{dj}\left(t-T_d^d\right)\right.\\ + K_{Vj}\ddot{E}_j\left(t-T_d^d\right) + K_{Pj}E_j\left(t-T_d^d\right)\\ + C_i\left(\theta\left(t-T_d^d\right),\dot{\theta}\left(t-T_d^d\right)\right) + G_i\left(\theta\left(t-T_d^d\right)\right)\end{aligned} \tag{8}$$

Here, T_d^d is the time delay (due to sampling time) in the discrete time distributed model-based control law. Note that it is expected that $T_d^d < T_d$ as the computational effort associated with the control law is now shared among the individual controllers. Hence, it is expected that the trajectory tracking performance of manipulator with the discrete time distributed model-based control law (8) is superior to that with the centralized, discrete-time model-based control law given in Eqn. (7).

6. Distributed Control for a 3R Planar Manipulator

Now we shall illustrate the control scheme given by Eqn. (4) implemented in the proposed distributed control architecture using a simple 3R planar manipulator. We consider a 3R planar manipulator for several reasons. First, most manipulators use

revolute joints, which result in nonlinearities and dynamic coupling. Second, a serial-link planar manipulator has maximum dynamic coupling between its links. Third, it is a simplest (in terms of degrees of freedom) manipulator with at least one intermediate link, and fourth, it is a simplest (in terms of degrees-of-freedom) redundant manipulator (considering only tool position in a plane without considering its orientation).

Figure 5 shows the block diagram of the control law (Eqn. (4)) implemented in the proposed distributed architecture. The communication links, along with the information exchange between the neighboring controllers are also shown.

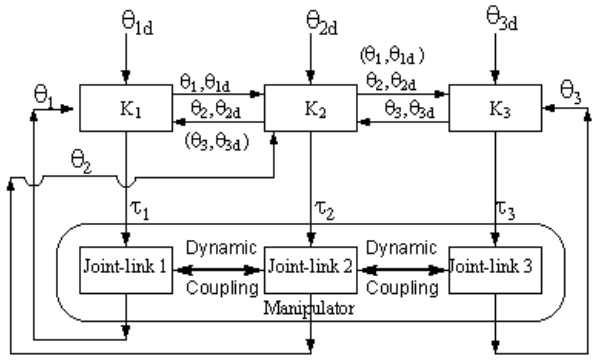


Fig. 5. Block diagram of the proposed model-based control of a 3R planar manipulator in the proposed distributed architecture

The i^{th} joint level controller K_i receives the desired trajectory (θ_{id}; $\dot{\theta}_{id}$; $\ddot{\theta}_{id}$) and the actual (or the current values of) (θ_i; $\dot{\theta}_i$; $\ddot{\theta}_i$) in the form of sensory feedback, as inputs, and computes the control input τ_i as given by the control law (4), for the i^{th} joint. Controllers communicate the values of corresponding joint variable (feedback) and desired joint variable (along with necessary derivatives not shown in the figure), that they received, to the immediate neighbors. The intermediate controller K_2 communicates the values of θ_1 (feedback it received via K_1, along with its first and second derivatives) and θ_{1d} (desired value it received via K_1) to K_3, and the values of θ_3 (feedback it received via K_3, along with the first and second derivatives) and θ_{3d} (desired value it received via K_3) to K_1. Now with this distributed multi-hop communication between the individual joint level controllers, each of them has all the necessary information to compute the corresponding control law. Finally, the controller K_i provides the control input τ_i to the i^{th} joint (the i^{th} joint-link agent) using Eqn. (4) or (8).

Remark 4. The control scheme provided in Fig. 5 may be implemented in hardware. The joint level controllers may be implemented on an embedded hardware with a provision for necessary communication between them. In the case of distributed control of a manipulator, unlike in a typical multi-agent/robotic system, it is possible to use wired communication between the joint level controllers. However, a detailed discussion on the hardware implementation is beyond the scope of this paper.

Computational cost and distribution effectiveness. Table 1 shows the number of addition (N_A), multiplication (N_M), along with the corresponding computational cost and the total computational cost (C_T) of computing dynamics at each joint level. We obtain a distribution effectiveness of 0.66 in this case, as against an ideal value of 1. The maximal computational cost with the distributed implementation now reduces from 944 units to 480 units, that is about 50% of the cost of centralized implementation. This implies that the sampling time of a discretized implementation of the control law in the distributed architecture is about half that of the centralized architecture. If the computational load were distributed equally among the joint level controllers, then the computational cost, and hence the sampling time, with the distributed implementation would have been 33% of that with the centralized implementation. Thus, though the model-based control law implemented in both centralized architecture (Eqn. (2)) and the proposed distributed architecture Eqn. (4) are theoretically identical, in reality, when the control law is implemented in discrete time, the trajectory tracking performance with the control law in distributed architecture is expected to be superior compared to that with the centralized architecture as $T_d^d = 0.5T_d$. If we carefully design the distributed control law such that $\eta_d = 1$, then we obtain $T_d^d = 0.33T_d$, the least possible sampling time. As shown in Table 2, the distribution effectiveness θ_d improves with degrees-of-freedom of the manipulator. It may be observed that manipulator control in the distributed architecture is more useful for the higher degrees-of-freedom manipulator due to higher computational cost of the centralized control law and better distribution effectiveness.

Tab. 1. Number of additions (N_A), multiplications (N_M), corresponding costs (C_A;C_M), and the total cost (C_T), involved in computation of the dynamics at each joint

Link#	NA	NM	CA	CM	CT
1	64	104	64	416	480
2	50	78	50	312	362
3	14	22	14	88	102
Total	128	204	128	816	944

Tab. 2. Distribution effectiveness with the degrees-of-freedom of planar manipulators

DOF	2	3	4	5	6
η_d	0.64	0.66	0.69	0.72	0.75

Remark 5. We may observe that the distributed control scheme based on Eqn. (4) presented here is based on the identification of a natural distributed nature of the manipulator dynamics itself and that of the model-based control law (2). The reduction in computational lead-time with the distributed control scheme is achieved purely because of the distribution of the computational effort among the joint level controllers, rather than the program optimization or operation optimization techniques that are used at the algorithmic level.

Tab. 3. Terms appearing multiple times in 3R manipulator dynamic equation

Sl.No	Repeating terms	Joint1	Joint2	Joint3
1	$-l_1\dot{\theta}_1^2 + s_1 g$	9	7	2
2	$l_1\ddot{\theta} + C_1 g$	10	7	2
3	$-l_2(\dot{\theta}_1 + \dot{\theta}_2)^2$	4	3	1
4	$l_2\left(\ddot{\theta}_1 + \ddot{\theta}_2\right)$	5	4	1
5	$l_2(\ddot{\theta}_1 + \ddot{\theta}_2 + \ddot{\theta}_3)$	2	2	1
6	$-l_3(\dot{\theta}_1 + \dot{\theta}_2 + \dot{\theta}_3)^2$	1	1	0

Reducing computational cost. With careful observation, we can identify several repetitive terms in the dynamics of a 3R planar manipulator. Such repetitive terms are shown in Table 3 along with the number of repetitions. For example, the term $(-l_1\dot{\theta}_1^2 + s_1 g)$ repeats ten times in the equation corresponding to the first joint, seven times in that corresponding to the second joint, and twice in that corresponding to the third joint.

Tab. 4. Number of addition and multiplication, corresponding computational cost after avoiding repetitive computation of terms shown in Table 3

Link ID	N_{OA}	N_{OM}	C_A	C_M	C_T
1	30	40	30	160	190
2	24	32	24	12	152
3	6	8	6	32	38
Total	60	80	60	320	30

Now if we compute each of the terms that are listed in Table 3 only once, we may further reduce the computation cost associated with dynamics, and hence, that of the control law, at each joint. Note that this reduction is achieved without neglecting any of the terms. Table 4 shows the number of addition, multiplication, along with the corresponding computational cost and total computational cost of computing dynamics at each joint level after this refinement. It may be observed that the computational cost at each joint level is now reduced by about 60% compared to that shown in Table 1. However, the distribution effectiveness η_d = 0:66 even in this case, indicating that this exercise of reducing computations by avoiding repeated computation of certain repetitive terms does not affect how the computation load is shared among the individual controllers.

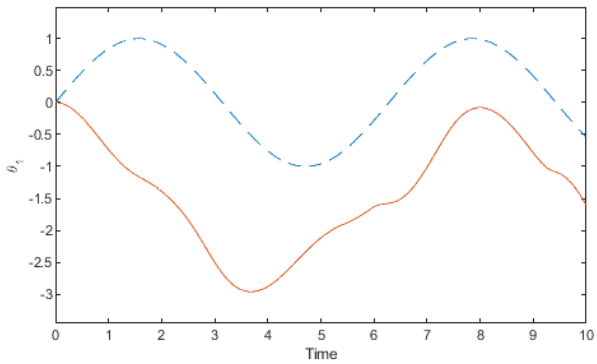
Remark 6. Apart from the reduction in computational overhead due to the natural distribution of the computational effort among the joint level controllers, we have achieved further reduction in the computational load here by identifying repetitive terms in the manipulator dynamics/control law. As demonstrated by the fact that the distribution effectiveness is unaffected by this exercise, this process of reduction in computational load is independent of the distributed property of the manipulator dynamics or the proposed distributed control scheme.

7. Results and Discussion

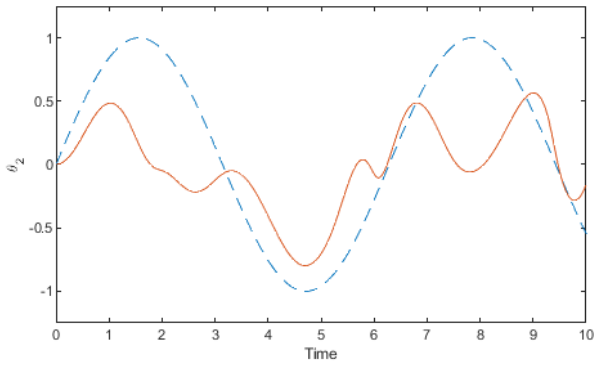
In this section, we present results of simulation experiments carried out in Matlab to illustrate and compare the trajectory tracking performance of the proposed control scheme with a simple decentralized PID control scheme. We also provide a discussion on the comparison of the proposed control scheme with that of the decentralized control schemes in general.

Simulation results. First, we present the results of simulation experiments. We have simulated the proposed distributed control scheme for a 3R planar manipulator using Matlab/Simulink. We considered a manipulator with m_1 = 10kg, m_2 = 10kg, and m_3 = 10kg, and l_1 = 5m; l_2 = 6m, and l_3 = 5m. We considered a sinusoidal trajectory as the desired trajectory to be followed by each of the joints.
Figures 6(a)-(c) show the trajectory tracking performance of the first, second, and third joints of the manipulator with the decentralized PID controller. Figures 7(a)-(c) show the trajectory tracking performance of the first, second, and third joints of the manipulator with the proposed distributed control. It can be observed that all the joints successfully track the respective desired trajectories with the proposed distributed controller. Though the performance with the decentralized PID control may be improved by tuning the controller gains, as observed earlier, due to the nonlinear nature of the manipulator dynamics, the performance level cannot be guaranteed to be uniform across the state-space. As expected, it was observed during the simulation experiments that the trajectories obtained with the proposed distributed control scheme were identical to that obtained with the centralized model-based control scheme.

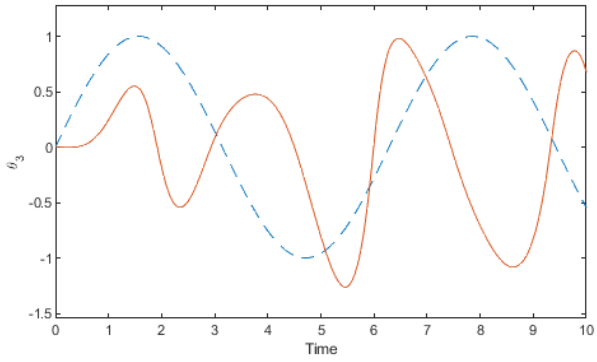
Distributed vs decentralized schemes. Now we provide an informal discussion comparing the proposed distributed (or centralized) control scheme with the



(a)



(b)

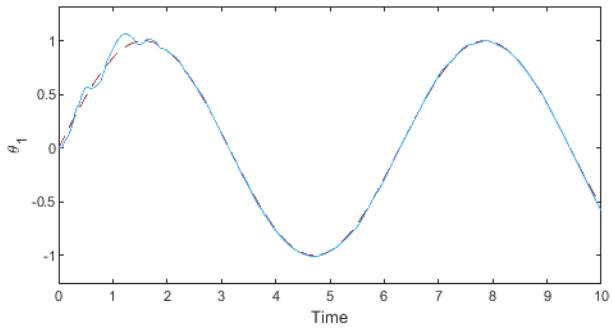


(c)

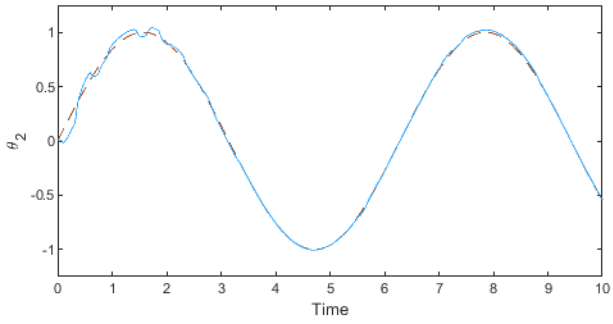
Fig. 6. Trajectory tracking performance of a) first, b) second joint, and c) third joint, of a 3R planar manipulator with independent joint PID controller

decentralized control scheme in general. The independent joint PID controller is probably the simplest control scheme reported in the literature in the decentralized control architecture. Each of such schemes leads to different trajectory tracking performance, which is expected to be better than that with the independent-joint PID control scheme. However, when the system model is fully available, it has been established theoretically that the trajectory tracking performance with the model-based nonlinear control is superior to that with any other control scheme which does not consider the model fully, particularly the coupled dynamics. In the case of control schemes in decentralized architecture inspite of using adaptive control or other techniques to account for unmodelled dynamics, there is no provision for accounting for the coupling dynamics between the links. Apart from this theoretically established fact of resulting in an inferior trajectory tracking performance compared to that with the model-based control (centralized or distributed), the decentralized control schemes proposed in the literature (unlike the simple independent joint PID scheme) may not be computationally very inexpensive.

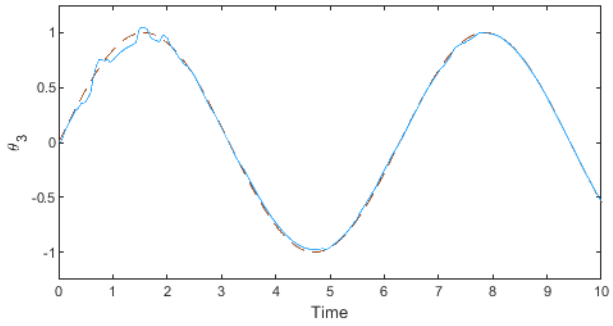
Thus, we may observe that though the decentralized schemes reported in the literature may have marginally lower computational overhead as compared to the proposed model-based control in the distributed architecture, their trajectory performance is expected to be inferior, at least in a theoretical sense and ideal situations, to that with the distributed scheme proposed in this work.



(a)



(b)



(c)

Fig. 7. Trajectory tracking performance of a) first, b) second joint, and c) third joint, of a 3R planar manipulator with the proposed control scheme

Though we consider the manipulator dynamics is known completely in this work, it may not be the case in reality. When the model is not known exactly, it is possible to use techniques such as adaptive control within the distributed control architecture. However, the focus of this paper is on control schemes implemented in a distributed architecture and establishing equivalence of the centralized and distributed architecture.

8. Conclusion

We proposed a distributed model-based control architecture for a manipulator. The classical model-based control law was used to demonstrate the proposed architecture by implementing it in a distributed manner. The distributed control scheme provided was shown to lead to a stable, linear, decoupled, second order error dynamics. The trajectory tracking performance with the proposed distributed model-based control scheme was observed to be identical to that with the centralized model-based control under ideal conditions. Further, with the proposed control scheme, the computational lead-time is shown to reduce considerably by distributing the computational effort among the individual controllers. This reduction in computational lead-time in turn was observed to improve the tracking performance when the control law is realized in discrete time. In contrast to the decentralized or independent-joint control schemes reported in the literature, the coupling dynamics are not neglected in the proposed distributed control scheme. Simulation results using Matlab were provided to demonstrate that the tracking performance of a 3R planar manipulator with the proposed distributed model based control scheme is superior to that with the decentralized PID control scheme.

A detailed comparison of the proposed distributed model-based control scheme with the decentralized control schemes presented in the literature in terms of the tracking performance and the computational time will be very useful. Some of the other directions for future work include design of distributed controller schemes to achieve a better distribution of the computational load, thereby improving the distribution effectiveness, and hence, reducing the computation lead time, devising a formal methodology for the design of the distributed control for a general serial link robot, experimental verification of the control scheme, etc.

AUTHORS

S. Soumya* – Instrumentation and Control Engineering, Manipal Institute of Technology, Manipal, Karnataka, India, e-mail: soumya5.subbu@gmail.com.

K. R. Guruprasad – Department of Mechanical Engineering, National Institute of Technology Karnataka, Surathkal, India.

*Corresponding author

REFERENCES

[1] F. Aghili, "Torque control of electric motors without using torque sensor". In: *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2007, 3604–3609, DOI: 10.1109/IROS.2007.4399145.

[2] P. Bohner and R. Luppen, "Reactive multi-agent based control of redundant manipulators". In: *Proceedings of International Conference on Robotics and Automation*, vol. 2, 1997, 1067–1072, DOI: 10.1109/ROBOT.1997.614276.

[3] S.-J. Cho, J. S. Lee, J. Kim, T.-Y. Kuc, P.-H. Chang and M. Jin, "Adaptive time-delay control with a supervising switching technique for robot manipulators", *Transactions of the Institute of Measurement and Control*, vol. 39, no. 9, 2017, 1374–1382, DOI: 10.1177/0142331216637118.

[4] P. Dasgupta, J. Baca, K. R. Guruprasad, A. Muñoz-Meléndez and J. Jumadinova, "The COMRADE System for Multirobot Autonomous Landmine Detection in Postconflict Regions", *Journal of Robotics*, vol. 2015, 2015, 1–17, DOI: 10.1155/2015/921370.

[5] K. R. Guruprasad, *Robotics: Mechanics and Control*, PHI LEARNING, 2019.

[6] K. R. Guruprasad and A. Ghosal, "Model Reference Learning Control for Rigid Robots". In: *Proceedings of the ASME Design Engineering Technical Conferences*, 1999, 1-9.

[7] K. R. Guruprasad and D. Ghose, "Automated Multi-Agent Search Using Centroidal Voronoi Configuration", *IEEE Transactions on Automation Science and Engineering*, vol. 8, no. 2, 2011, 420–423, DOI: 10.1109/TASE.2010.2072920.

[8] "Complete Coverage of an Initially Unknown Environment by Multiple Robots using Voronoi Partition". K. R. Guruprasad, Z. Wilson and P. Dasgupta, https://pdfs.semanticscholar.org/2981/639efaf2931e24b1d0be8fd8cba6aeb18e35.pdf. Accessed on: 2020-05-28.

[9] M. Hashimoto, "Robot motion control based on joint torque sensing". In: *1989 International Conference on Robotics and Automation Proceedings*, 1989, 256–261, DOI: 10.1109/ROBOT.1989.99998.

[10] T. C. Hsia and L. S. Gao, "Robot manipulator control using decentralized linear time-invariant time-delayed joint controllers". In: *Proceedings of IEEE International Conference on Robotics and Automation*, 1990, 2070–2075, DOI: 10.1109/ROBOT.1990.126310.

[11] S.-H. Hsu and L.-C. Fu, "A fully adaptive decentralized control of robot manipulators", *Automatica*, vol. 42, no. 10, 2006, 1761–1767, DOI: 10.1016/j.automatica.2006.05.012.

[12] H. Jia, W. Zhuang, Y. Bai, P. Fan and Q. Huang, "The Distributed Control System Of a Light Space Manipulator". In: *2007 International Conference on Mechatronics and Automation*, 2007, 3525–3530, DOI: 10.1109/ICMA.2007.4304131.

[13] L. Jin, S. Li, J. Yu and J. He, "Robot manipulator control using neural networks: A survey", *Neurocomputing*, vol. 285, 2018, 23–34, DOI: 10.1016/j.neucom.2018.01.002.

[14] K. Kosuge, H. Takeuchi and K. Furuta, "Motion control of a robot arm using joint torque sensors", *IEEE Transactions on Robotics and Automation*, vol. 6, no. 2, 1990, 258–263, DOI: 10.1109/70.54743.

[15] S. Li, H. Wang and M. U. Rafique, "A Novel Recurrent Neural Network for Manipulator Control With Improved Noise Tolerance", *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 5, 2018, 1908–1918, DOI: 10.1109/TNNLS.2017.2672989.

[16] M. Liu, "Decentralized PD and Robust Nonlinear Control for Robot Manipulators", *Journal of Intelligent and Robotic Systems*, vol. 20, no. 2, 1997, 319–332, DOI: 10.1023/A:1007968513506.

[17] M. Liu, "Decentralized control of robot manipulators: nonlinear and adaptive approaches", *IEEE Transactions on Automatic Control*, vol. 44, no. 2, 1999, 357–363, DOI: 10.1109/9.746266.

[18] A. Mori, F. Naya, N. Osato and T. Kawaoka, "Multiagent-based distributed manipulator control". In: *1996 IEEE/SICE/RSJ International Conference on Multisensor Fusion and Integration for Intelligent Systems*, 1996, 289–296, DOI: 10.1109/MFI.1996.572190.

[19] P. Poignet and M. Gautier, "Nonlinear model predictive control of a robot manipulator". In: *Proceedings of 6th International Workshop on Advanced Motion Control*, 2000, 401–406, DOI: 10.1109/AMC.2000.862901.

[20] L. E. Pfeffer, O. Khatib and J. Hake, "Joint torque sensory feedback in the control of a PUMA manipulator", *IEEE Transactions on Robotics and Automation*, vol. 5, no. 4, 1989, 418–425, DOI: 10.1109/70.88056.

[21] H. Seraji, "Decentralized adaptive control of manipulators: theory, simulation, and experimentation", *IEEE Transactions on Robotics and Automation*, vol. 5, no. 2, 1989, 183–201, DOI: 10.1109/70.88039.

[22] J.-E. Slotine and L. Weiping, "Adaptive manipulator control: A case study", *IEEE Transactions on Automatic Control*, vol. 33, no. 11, 1988, 995–1003, DOI: 10.1109/9.14411.

[23] S. Soumya and K. R. Guruprasad, "Model-based distributed cooperative control of a robotic manipulator". In: *2015 IEEE International WIE Conference on Electrical and Computer Engineering (WIECON-ECE)*, 2015, 313–316, DOI: 10.1109/WIECON-ECE.2015.7443926.

[24] M. W. Spong and M. Vidyasagar, *Robot dynamics and control*, Wiley, 1989.

[25] T. Tsuji, S. Nakayama and K. Ito, "Distributed feedback control for redundant manipulators based on virtual arms". In: *Proceedings ISAD 93: International Symposium on Autonomous Decentralized Systems*, 1993, 143–149, DOI: 10.1109/ISADS.1993.262710.

[26] Z.-J. Yang, Y. Fukushima and P. Qin, "Decentralized Adaptive Robust Control of Robot Manipulators Using Disturbance Observers", *IEEE Transactions on Control Systems Technology*, vol. 20, no. 5, 2012, 1357–1365, DOI: 10.1109/TCST.2011.2164076.

[27] J. Yim and J. H. Park, "Nonlinear H∞ control of robotic manipulator". In: *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, vol. 2, 1999, 866–871.

[28] J. S. Yuan, "Closed-loop manipulator control using quaternion feedback", *IEEE Journal on Robotics and Automation*, vol. 4, no. 4, 1988, 434–440, DOI: 10.1109/56.809.