Robotics and Autonomous Systems 87 (2017) 226–236

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

Flat control of industrial robotic manipulators

Elisha D. Markus [a,*], John T. Agee [b], Adisa A. Jimoh [c]

[a] Department of Electrical, Electronic and Computer Systems, Central University of Technology, Free State, South Africa
[b] Department of Electrical Engineering, University of Kwazulunatal, Durban, South Africa
[c] Department of Electrical Engineering, Tshwane University of Technology, Pretoria, South Africa

HIGHLIGHTS

- Computational burden of high order robotic control significantly reduced.
- Computation of flat output for 6-DOF robotic manipulator presented.
- Comparison between PID and Flatness based control of ABB IRB140 robot provided.
- Tracking control of 6-DOF ABB IRB140 robot in Joint space and Task space presented.

ARTICLE INFO

Article history:
Received 11 April 2015
Received in revised form 13 September 2016
Accepted 6 October 2016
Available online 24 October 2016

Keywords:
Industrial robots
Six degrees of freedom robot
Differential flatness
Trajectory planning
Tracking control

ABSTRACT

A new approach to tracking control of industrial robot manipulators is presented in this paper. The highly coupled nonlinear dynamics of a six degrees of freedom (6-DOF) serial robot is decoupled by expressing its variables as a function of a flat output and a finite number of its derivatives. Hence the derivation of the flat output for the 6-DOF robot is presented. With the flat output, trajectories for each of the generalized coordinates are easily designed and open loop control is made possible. Using MATLAB/Simulink S-functions combined with the differential flatness property of the robot, trajectory tracking is carried out in closed loop by using a linear flat controller. The merit of this approach reduces the computational complexity of the robot dynamics by allowing online computation of a high order system at a lower computational cost. Using the same processor, the run time for tracking arbitrary trajectories is reduced significantly to about 10 s as compared to 30 min in the original study (Hoifodt, 2011). The design is taken further by including a Jacobian transformation for tracking of trajectories in cartesian space. Simulations using the ABB IRB140 industrial robot with full dynamics are used to validate the study.

1. Introduction

The development of industrial robots has been advanced in the last decade. This is mainly due to increase in the complexity of tasks that they execute. The controller which is a major component of robots has received a lot of attention from robotic researchers. This is because it has a direct impact on their performance and could inhibit their deployability and applicability in certain areas [1–3]. Many control techniques have been proposed for modern industrial robot manipulators including the classical PID, Computed torque, feedback linearization, inverse dynamics, neurofuzzy, model predictive control etc. More recently, robot control methods have been model based which simply relies on the mathematical model of the robot.

Model based control has been applied in trajectory tracking tasks [4–6]. However, the computational requirements of these model based systems are quite high as it is required to solve very large equations in their controls. Despite fast computer processor speeds, effective control algorithms become very computationally expensive with high run times and in some cases impossible to achieve. Most researchers in dealing with trajectory robot tracking applications have had to work with lower order robotic dynamics such as in wheeled robots [7–9], parallel robots [10] and underactuated robots [1,8,11] in proposing their control algorithms. Underactuating robotic manipulators reduces the order from a high degree of freedom (DOF) to a lower one. As the number of degree of freedom increases, the computational complexity of computing the control also increases. We refer to chapter 6 of [12] where the computational complexity of the 6-dof robot dynamics is discussed.

In this study, a 6-dof robot is modeled using the Newton–Euler approach. The computations done with the MAPLE software resulted in very long dynamic equations, literally running into several pages. The mere size of the equations can be a challenge to the control engineer in terms of the computing time. The equations

* Corresponding author.
E-mail address: emarkus@cut.ac.za (E.D. Markus).

http://dx.doi.org/10.1016/j.robot.2016.10.009
0921-8890/

are also nonlinear which adds to the burden. In these scenarios, classical control techniques fail due to the fact that they require long times for solutions to be obtained. This study presents a technique of tracking a 6 DOF without the use of underactuation or such other assumptions that would have first reduced the complexity of the problem.

One nonlinear control strategy that is gaining popularity among robotic researchers is the differential flatness based control [13–18]. It has been applied to control mobile robots [7,9,19], UAVs, UGVs [20], flexible robots [21], underactuated planar robot [11,22–24] and so on. Differential flatness is known to be well suited for the problem of trajectory generation and tracking [18,19,25]. With differential flatness, the trajectories (position, velocity, acceleration and jerk) of a nonlinear system can be easily interpolated by defining a smooth curve with initial and final conditions. The state and control variables can then be reconstructed without having to integrate the system equations [18].

This paper focuses on using this strategy for trajectory control of the ABB IRB140 6-DOF industrial robot with full dynamics. This is the main contribution of this paper. Trajectory control is carried out in open loop and then tracking is done using feedback in a flat controller. The control design is implemented both in joint space and task space. Of particular interest is the performance of tracking in task space since there is generally a problem of lack of synchronization between motion in the end effector and that in joint space. A small tracking error in the joint space can easily be magnified in the operational space. The tracking control is simulated using the MATLAB/Simulink software environment. The study is an extension of earlier results on differential flatness based control by the authors [26].

The paper is organized as follows: Section 2 describes the dynamic model of an n-dof robot with particular reference to the ABB IRB140 robot. Section 3 discusses differential flatness analysis of the robot and shows the flat output computation. Section 4 introduces trajectory planning and the flatness based controller design. Section 5 discusses simulation and results of the flatness based trajectory control for the IRB140 model. The paper is concluded in Section 6.

2. Modeling of 6-dof robot

First we start by modeling the kinematics of the robot. The robot kinematics describes motion of the joints without recourse to the forces causing them. We present the forward and inverse kinematics of the Irb140 robot.

2.1. Kinematic modeling ABB Irb140 robot

The forward kinematics is used to find the position of the manipulator given the joint positions. This is easy once the DH parameters of the robot is known. For this study, we use the DH parameters as given in [27]. Given the actual end effector pose of the robot, the inverse kinematics solves for the corresponding joint positions. The inverse kinematics problem has many solutions so it is not always unique: the same end effector pose can be reached in several configurations, corresponding to distinct joint position vectors. For robot manipulators with six DOF, with the last three joints intersecting at a wrist, such as the Irb140 robot, it is common practice to decouple the inverse kinematics problem into two simpler problems. The problem is defined in terms of the inverse position kinematics and inverse orientation kinematics. Fig. 1 describes the Kinematic structure and frame assignments of the ABB IRB 140 robot.

The DH parameters of the IRB140 robot as described by [27] are shown in Table 1:

Table 1
DH parameters of Irb140 Robot.

Axis i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	d_1	θ_1
2	90	a_1	0	θ_2
3	0	a_2	0	θ_3
4	90	0	d_4	θ_4
5	−90	0	0	θ_5
6	90	0	d_6	θ_6

where $d_1 = 352$ mm, $a_1 = 70$ mm, $a_2 = 360$ mm, $d_4 = 380$ mm, $d_6 = 65$ mm.

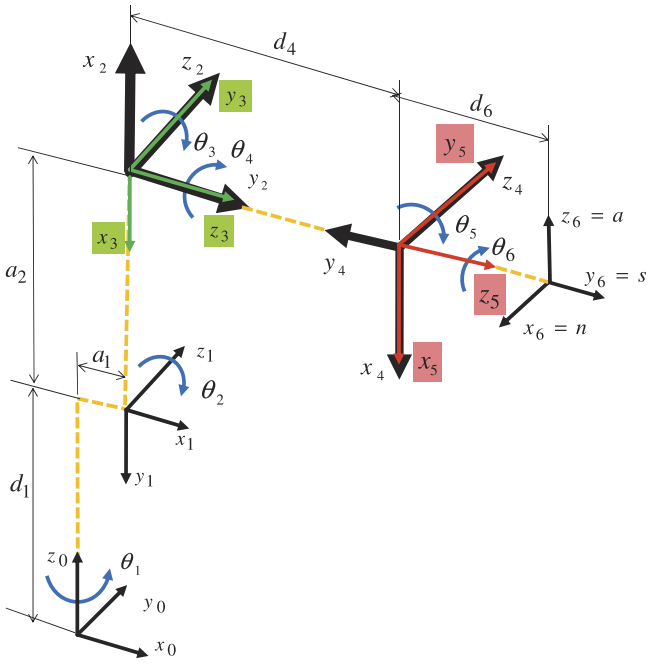

Fig. 1. Kinematic structure and frame assignments of the ABB IRB 140 robot.

A geometrical solution for the inverse kinematics as derived by [10] is described here: The Transformation matrix from frame 6 to frame 1 with respect to the base frame 0 defines the cartesian position and orientation of the end effector as:

$$ {}^{0}_{6}T = \begin{bmatrix} r_{11} & r_{12} & r_{13} & P_{xw} \\ r_{21} & r_{22} & r_{23} & P_{yw} \\ r_{31} & r_{32} & r_{33} & P_{zw} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1} $$

where:

$$ \begin{aligned} r_{11} &= c_1c_{23}(c_4c_5c_6 - s_4s_6) - c_1s_5s_{23}c_6 + s_1(s_4c_5c_6 + c_4s_6) \\ r_{12} &= -c_1c_{23}c_4c_5c_6 + c_1s_{23}s_5s_6 - s_1s_4c_5s_6 \\ r_{13} &= c_1c_{23}c_4s_5 + c_1s_{23}c_5 + s_1s_4s_5 \\ r_{21} &= s_1s_{23}(c_4c_5c_6 - s_4s_6) - s_1s_{23}s_5c_6 - c_1(s_4c_5c_6 + c_4s_6) \\ r_{22} &= -s_1c_{23}c_4c_5c_6 + s_1s_{23}s_5s_6 - c_1s_4c_5s_6 \\ r_{23} &= s_1c_{23}c_4c_5c_6 + s_1s_{23}s_5 - c_1s_4s_5 \\ r_{31} &= s_{23}(c_4c_5c_6 - s_4s_6) + c_{23}s_5c_6 \\ r_{32} &= -c_6s_{23}c_4c_5 - c_{23}s_5s_6 \\ r_{33} &= s_{23}c_4c_5 - c_{23}c_5 \\ P_{xw} &= c_1s_{23}d_4 + c_1c_2a_2 + c_1a_1 \\ P_{yw} &= s_1s_{23}d_4 + s_1c_2a_2 + s_1a_1 \\ P_{zw} &= -c_{23}d_4 + s_2a_2 + d_1. \end{aligned} \tag{2} $$

P_{xw}, P_{yw}, P_{zw} describes the position of the wrist center. $c_1 = cos(q_1)$, $s_{12} = sin(q_1 + q_2)$, etc.

The solution for the first angle q_1 from Fig. 2 is:

$$ q_1 = atan(P_{yw}, p_{xw}). \tag{3} $$

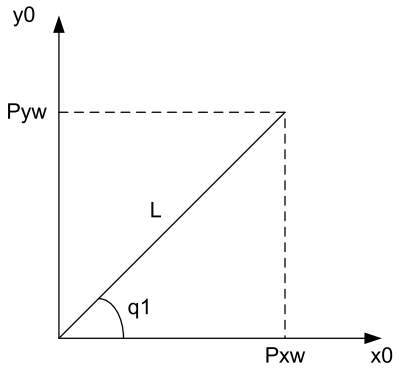

Fig. 2. Projection of the wrist center onto the *xy* plane.

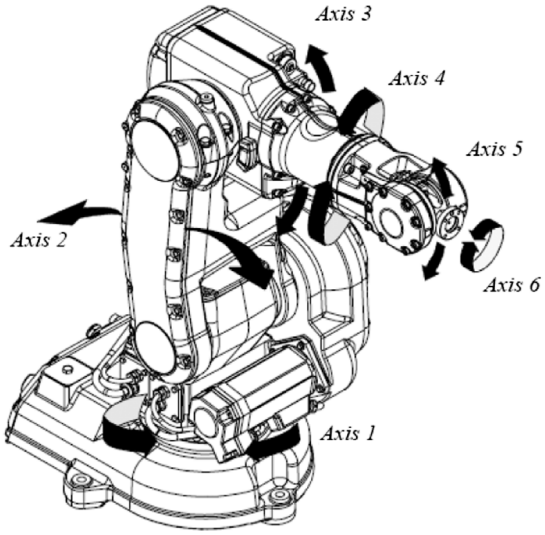

Fig. 3. ABB Irb140 Robot with six axes [29].

The expressions for the remaining joints q_2 to q_6 will be included in the Appendix. With the inverse kinematics, we can determine the joint positions required for joint controls given a reference Cartesian trajectory.

2.2. Dynamic modeling of ABB Irb140 robot

For dynamic modeling of industrial robots with n degrees of freedom, two methods exist: The Euler–Lagrange and the Newton–Euler method. While the former approach is energy based, the latter analyzes the forces between each of the links in a recursive manner. It has been shown that the Newton–Euler method is more suited for modeling higher degree of freedom robots since it is faster and provides a more accurate model [28]. In this study, the Newton–Euler method was used to derive the dynamic model of the robot which takes the form of:

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q}) = \mathbf{K}(\mathbf{q}, \dot{\mathbf{q}})\mathbf{u} \tag{4}$$

where $\mathbf{q}(\mathbf{t}) \in \Re^n$ is the vector of angular joint positions called the generalized coordinates. $\mathbf{u}(\mathbf{t}) \in \Re^n$ are the control inputs or forces driving the joints. $\mathbf{M}(\mathbf{q})$ is a positive definite $n \times n$ invertible matrix representing the inertia forces [12]. $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ is an n dimensional vector of coriolis and centrifugal terms and $\mathbf{G}(\mathbf{q})$ is an n dimensional vector representing the gravitational forces. $\mathbf{K}(\mathbf{q}, \dot{\mathbf{q}})$ is the matrix characterizing the actuating forces. It is also assumed that for an n-dof robot, Fig. 3 $dim(\mathbf{q}) = dim(\mathbf{u})$ and $rank(\mathbf{K}) = n$.

It is important to express (4) in a first order nonlinear form to facilitate its representation for flatness analysis. The equation can be written in the form

$$\dot{\mathbf{q}} = \mathbf{f}(\mathbf{q}, \mathbf{u}). \tag{5}$$

Hence based on (5), we can write (4) as:

$$\ddot{\mathbf{q}} = -\mathbf{M}(\mathbf{q})^{-1}(\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q})) + \mathbf{M}(\mathbf{q})^{-1}\mathbf{K}(\mathbf{q}, \dot{\mathbf{q}})\mathbf{u}. \tag{6}$$

For a 6-dof robot, Eq. (6) has six second-order differential equations. This can be transformed to twelve first order differential equations. Setting:

$$\begin{aligned} &x_1 = q_1, x_7 = \dot{q_1} \\ &x_2 = q_2, x_8 = \dot{q_2} \\ &\qquad\vdots \\ &x_5 = q_5, x_{11} = \dot{q_5} \\ &x_6 = q_6, x_{12} = \dot{q_6}. \end{aligned} \tag{7}$$

The first order form is hereby expressed by:

$$\begin{aligned} &\dot{x_1} = x_7, \dot{x_7} = f_7(x, u) = \ddot{q_1} \\ &\dot{x_2} = x_8, \dot{x_8} = f_4(x, u) = \ddot{q_2} \\ &\qquad\vdots \\ &\dot{x_6} = x_{12}, \dot{x_{12}} = f_{12}(x, u) = \ddot{q_6}. \end{aligned} \tag{8}$$

The equation in states space is now of the form

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) + \mathbf{g}(\mathbf{x})\mathbf{u} \tag{9}$$

and the output vector is given by:

$$\mathbf{y} = \mathbf{h}(\mathbf{x}) \tag{10}$$

$\mathbf{f}(\mathbf{x}) : \Re^{2n} \to \Re^{2n}$, $\mathbf{g}(\mathbf{x}) : \Re^n \to \Re^{2n}$. f maps $\Re^{2n}$ to $\Re^{2n}$, $\mathbf{g}$ is a smooth matrix function and $\mathbf{h}$ is a smooth function (defined in Eq. (12)). This means that the functions are continuously differentiable for sufficient number of times. These vectors are defined as:

$$\begin{aligned} \mathbf{f}(\mathbf{x}) &= \begin{bmatrix} \mathbf{x_2} \\ \mathbf{M}(\mathbf{x_1})^{-1}(-\mathbf{C}(\mathbf{x_1}, \mathbf{x_2})\mathbf{x_2} - \mathbf{G}(\mathbf{x_1})) \end{bmatrix} \\ \mathbf{g}(\mathbf{x}) &= \begin{bmatrix} 0_{n\times n} \\ \mathbf{M}(\mathbf{x_1})^{-1} \end{bmatrix}. \end{aligned} \tag{11}$$

Eq. (11) depicts a 2nd order system with 12 state variables. The outputs to be controlled are a vector of the joint angles of the robot given by:

$$\mathbf{y} = \mathbf{h}(\mathbf{x}) = \mathbf{C}\mathbf{x} \tag{12}$$

where C is a matrix defined by $\mathbf{C} = \begin{bmatrix} I_{n\times n} & 0_{n\times n} \end{bmatrix}$.

3. Differential flatness analysis of an n-dof robot arm

Differential flatness is derived from differential geometry which allows us to simplify high dimensional computational problems of differential manifolds. The merit of the approach is to reduce the computational complexity of the dynamics of the system by defining a flat output which enables us to characterize the dynamics of whole robot as trivial. For an n-dof robot with n actuators to be flat, all the n states and inputs of the robot must be defined in terms of the flat output and a finite number of its derivatives. We compute the flat output in the next subsection.

If the dynamic equation given in Eq. (4) is equivalent to equations in terms of the flat output, then the robot is said to be Lie–Backlund (LB) equivalent (see [18]) to a linear system by endogenous feedback [30,31]. Once the dynamic equations are completely linearized, then the trajectories of the n-dof robot can

now be planned and controlled with much ease. This property of equivalence makes differential flatness well suited for motion planning in complex robotic manipulators.

From the affine system of (9), the whole system can be expressed in terms of its flat output such that the generalized output is **y** of dimension n. **y** and its successive derivatives are independent and the system variables **x** and **u** are expressed in terms of the generalized coordinates **y** and its successive derivatives to a finite number. That means:

$$(y_1 \ldots\ldots\ldots\ldots y_m) = \phi(x, u, \ldots\ldots u^{(l)})$$

and

$$\mathbf{x} = \phi_0(\mathbf{y}, \dot{\mathbf{y}}, \ddot{\mathbf{y}} \ldots\ldots, \mathbf{y}^{(s)}), \quad \mathbf{u} = \phi_1(\mathbf{y}, \dot{\mathbf{y}}, \ddot{\mathbf{y}} \ldots\ldots, \mathbf{y}^{(s+1)})$$

where s and l are positive integers.

The vector **y** can be represented by:

$$\mathbf{y} = (x_1, x_2, x_3, x_4, x_5, x_6)^T. \tag{13}$$

We now show the mathematical derivation of the flat output for the 6-DOF robot.

3.1. Flat output computation

The dynamics of the robot represented by Eq. (4) reads

$$\mathbf{M}(\mathbf{x})\ddot{\mathbf{x}} + \mathbf{C}(\mathbf{x}, \dot{\mathbf{x}})\dot{\mathbf{x}} + \mathbf{G}(\mathbf{x}) = \mathbf{u}$$

where **u** is the vector of exterior forces and torques. We can also write the equation in the form of:

$$\begin{aligned} &\mathbf{x_2} = \dot{\mathbf{x}}_1 \\ &\mathbf{u} = \mathbf{M}(\mathbf{x_1})\dot{\mathbf{x}}_2 + \mathbf{C}(\mathbf{x_1}, \mathbf{x_2})\mathbf{x_2} + \mathbf{G}(\mathbf{x_1}). \end{aligned} \tag{14}$$

Applying the algorithm based on the Smith decomposition [18], after eliminating **u** from the above equation, the system reads

$$\dot{\mathbf{x}}_1 - \mathbf{x_2} = \mathbf{0}$$

or

$$\left(\frac{d}{dt} - 1\right)\begin{pmatrix}\mathbf{x_1}\\ \mathbf{x_2}\end{pmatrix} = \mathbf{0} \tag{15}$$

which is a linear equation.

Right-multiplying the matrix of dynamics $\begin{pmatrix}\frac{d}{dt} & -1\end{pmatrix}$ by $\begin{pmatrix}0 & 1\\ -1 & \frac{d}{dt}\end{pmatrix}$ gives $\begin{pmatrix}1 & 0\end{pmatrix}$ which is the desired Smith decomposition. Therefore, keeping the second column of the matrix $\begin{pmatrix}0 & 1\\ -1 & \frac{d}{dt}\end{pmatrix}$, i.e. $\begin{pmatrix}1\\ \frac{d}{dt}\end{pmatrix}$ gives the relation

$$\begin{pmatrix}\mathbf{x_1}\\ \mathbf{x_2}\end{pmatrix} = \begin{pmatrix}1\\ \frac{d}{dt}\end{pmatrix}\mathbf{y} \tag{16}$$

and

$$\mathbf{y} = \mathbf{x_1} \tag{17}$$

which means that $\mathbf{x_1}$ is the flat output.

The flat output **y** usually has a physical meaning. In this case, it is the vector of angular displacements at each joint of the robot.

3.2. Lie derivatives

In order to establish the flatness and exact linearization by feedback for the fully actuated robot, we employ the Lie derivative function to fulfill certain conditions for the affine equation of (9).

The Lie derivative of a scalar function $h(x)$ with respect to a vector field f is defined by [32,33] $L_f h = \Delta h f$.

$L_g L_f^{k-1} h(x) = 0 \quad \forall x$ in a neighborhood of x, $k = 1 \ldots\ldots\ldots r-1$ and

$$L_g L_f^{r-1} h(x^0) \neq 0. \tag{18}$$

From the definition in (18), r is the relative degree of the system. Differentiating the output y, using Lie derivatives, we have,

$$\begin{aligned} &y^{(k)} = L_f^k h(x),\ k = 1 \ldots\ldots r-1 \\ &y^{(k)} = L_f^k h(x) + L_g L_f^{k-1} h(x) u, \quad k = r. \end{aligned} \tag{19}$$

The derivatives of the output will be:

$$\begin{aligned} \dot{y} &= \frac{\partial h}{\partial x}\dot{x} = \frac{\partial h}{\partial x}(f + gu) = L_f h(x) + L_g h(x) u \\ \ddot{y} &= \frac{\partial L_f h}{\partial x}(f + gu) = L_f^2 h(x) + L_g L_f h(x) u \end{aligned} \tag{20}$$

using the output vector from Eq. (12) we have

$$\begin{aligned} &y = h(x) = \mathbf{x} = L_f^0 h(x) \\ &\dot{y} = \dot{\mathbf{x}} \\ &\ddot{y} = -\mathbf{m^{-1}}(\mathbf{cx}^* + \mathbf{g}) + \mathbf{m^{-1}}\mathbf{u} \end{aligned} \tag{21}$$

where

$$\mathbf{cx}^* = \begin{bmatrix} c_1 x_7 & c_2 x_8 & c_3 x_9 & c_4 x_{10} & c_5 x_{11} & c_6 x_{12}\end{bmatrix}^T$$
$$\mathbf{g} = \begin{bmatrix} g_1 & g_2 & g_3 & g_4 & g_5 & g_6\end{bmatrix}^T$$

$\mathbf{m^{-1}}$ is the matrix describing the inverse of the inertia matrix. $\mathbf{cx}^*$ represents the vector of coriolis terms and **g** is the gravity vector.

Hence

$$\begin{aligned} &y = h(x) = L_f^0 h(x) \\ &\dot{y} = L_f^1 h(x) + L_g h(x) u = L_f^1 h(x), \quad L_g h(x) = 0 \\ &\ddot{y} = L_f^2 h(x) + L_g L_f^1 h(x) u = v. \end{aligned} \tag{22}$$

After differentiating the output twice, we obtain an input on the RHS of Eq. (22). Hence the relative degree r for the robot is 2. The manipulator described in (4) has $2n$ states and n inputs actuating every joint of the manipulator. If the manipulator is fully actuated, then we can have up to n outputs such that $y_i = q_i$, $i = 1, \ldots, n$ where each of these outputs has a relative degree of 2. The total sum of the relative degree for the manipulator for the n outputs is given as $2n$. This is the same as the dimension of the manipulator states. This implies that the whole state (of dimension $2n$) and input can be recovered by y and its derivatives up to the second order. Hence the exact linearization by feedback and flatness for the fully actuated robot is established [8,18,34]. Flatness in this case is equivalent to the well-known input/output decoupling property without zero dynamics and controlled torque method.(ref). Using a chain of two integrators we can design a linear system of the form

$$\ddot{y} = v \tag{23}$$

where v is given by:

$$v = \alpha + \beta u$$

and

$$\begin{aligned} &\alpha = -\mathbf{m^{-1}}(\mathbf{cx}^* + \mathbf{g}) \\ &\beta = -\mathbf{m^{-1}} \end{aligned} \tag{24}$$

v can be designed using pole placement to stabilize the loop.

$$v = \ddot{x}_{1ref} - k_1 \dot{x}_1{}^* - k_0 x_1{}^* \tag{25}$$

where $x^* = x_1 - x_{1ref}$.

To illustrate a global change of coordinates, we represent the generalized coordinates y by a new variable z, from Eq. (21), we

define $y = z$ as the flat output. The state vector transformation is thereby,

$$z_1 = \mathbf{x_1}$$
$$z_2 = \Delta_{z1} f = \mathbf{x_2} \tag{26}$$

$\mathbf{x_1}$ and $\mathbf{x_2}$ represent the vector of the generalized coordinates and its derivative respectively.

The set of linear equations of the 2nd order is given by:

$$\dot{z}_1 = z_2$$
$$\dot{z}_2 = v \tag{27}$$

and the input transformation $\mathbf{u}$ becomes

$$\mathbf{u} = \mathbf{K^{-1}}(\mathbf{x_1}, \dot{\mathbf{x_1}})(\mathbf{M}(\mathbf{x_1})\ddot{\mathbf{x_1}} + \mathbf{C}(\mathbf{x_1}, \dot{\mathbf{x_1}}) + \mathbf{G}(\mathbf{x_1})). \tag{28}$$

The vector u can be expanded for the individual joints as follows:

$$\begin{aligned}
u_1 &= m_1(x_1)\ddot{x}_1 + C_1(x_1, \dot{x}_1) + G_1(x_1)\\
u_2 &= m_2(x_1)\ddot{x}_1 + C_2(x_1, \dot{x}_1) + G_2(x_1)\\
u_3 &= m_3(x_1)\ddot{x}_1 + C_3(x_1, \dot{x}_1) + G_3(x_1)\\
u_4 &= m_4(x_1)\ddot{x}_1 + C_4(x_1, \dot{x}_1) + G_4(x_1)\\
u_5 &= m_5(x_1)\ddot{x}_1 + C_5(x_1, \dot{x}_1) + G_5(x_1)\\
u_6 &= m_6(x_1)\ddot{x}_1 + C_6(x_1, \dot{x}_1) + G_6(x_1).
\end{aligned} \tag{29}$$

It can be seen that these control inputs are all expressed in terms of the flat output x_1. The variables of Eq. (29) are further defined in the Appendix.

3.3. Trajectory planning

Trajectory planning for nonlinear systems requires an iterative solution by numerical methods to find control such that the initial and final time maneuvers for the system dynamics are satisfied. This process is often plagued by input saturation, nonlinearities and singularities. However, with the flat output: for the variables $x_1(t)$ given by the curve $t \mapsto x_1(t)$, there exist a trajectory

$$t \mapsto \begin{pmatrix} x(t) \\ u(t) \end{pmatrix} = \begin{pmatrix} \phi_0(x_1(t), \dot{x}_1(t), \ddot{x}_1(t) \ldots\ldots, {x_1}^{(q)}(t)) \\ \phi_1(x_1(t), \dot{x}_1(t), \ddot{x}_1(t) \ldots\ldots, {x_1}^{(q+1)}(t)) \end{pmatrix} \tag{30}$$

that satisfies the robot manipulator system equations as follows;

$$t \mapsto \begin{pmatrix} x(t) \\ u(t) \end{pmatrix} = \begin{pmatrix} x_1(t), \dot{x}_1(t), \ddot{x}_1(t) \\ M(x)\ddot{x}_1(t) \end{pmatrix} = \begin{pmatrix} x_1(t), \dot{x}_1(t), \ddot{x}_1(t) \\ M(x_1)v(t) \end{pmatrix}. \tag{31}$$

These trajectories are obtained without integrating any system of differential equations. The robot states $x(t)$ and inputs $u(t)$ are parameterized by the flat output $x_1(t)$ using (31). Hence the motion planning problem can be defined as finding a trajectory $t \mapsto x^*(t)$ and $t \mapsto u^*(t)$ satisfying $\dot{x}^*(t) = f(x^*(t), u^*(t))$ from an initial state $x_0 \in \mathrm{x}_1$ to a final state $x_T \in \mathrm{x}_1$ (where $T > 0$) such that

$$x^*(0) = x_0, \quad x^*(T) = x_T.$$

To track the reference trajectory with added disturbances or uncertainty, we define a function [18] $f_d \in T\mathrm{x}_1$ given a reference trajectory $t \mapsto x^*(t)$ of (x, f). We find a feedback law $x \mapsto u(x)$ and the error $x - x^*$ denoted by e such that

$$\dot{e}(t) = f_d(e(t) + {x_1}^*(t),\ u(e(t) + {x_1}^*(t))) - f({x_1}^*(t), u^*(t)) \tag{32}$$

is asymptotically stable for all disturbances.

There are many methods of generating these trajectories. They include, Bezier interpolation, fourier series and polynomials. Newton interpolation is used in this study to generate the coefficients of the flat output polynomials.

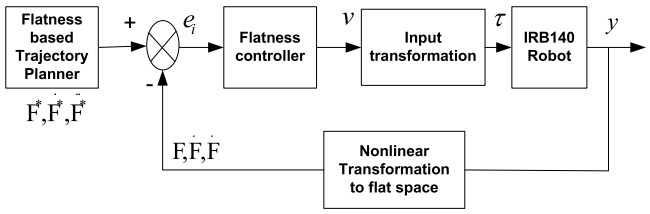

Fig. 4. Flatness based Controller Design in joint space.

4. Flatness based controller design

The robot control problem is seen as a determination of torques or forces generated by the actuators for the purpose of accomplishing a desired task. For the robot in study, we desire to accurately execute a planned trajectory using the end effector (e.g. in laser welding applications). These kinds of tasks require robust controllers to achieve minimum errors. The design of the controller is model based. Hence with the computed dynamics described in Section 2 and the flat output of the robot, the controller parameters are determined. The inertias, gravity, coriolis and centrifugal forces are all considered in the design. The control is done in two stages: first is the open loop control which involves the generation of nominal flat feedforward trajectories. These explicit trajectories allow for a more precise tracking of the robot. In the second stage, we take advantage of flatness property of the robot in designing a feedback linearizing control that will stabilize the robot in the presence of disturbances. The approach to flatness based control is known as the two degrees of freedom control [35]. One aspect deals with feedforward nominal control using trajectories and the second facet deals with feedback in the presence of disturbances. We consider both scenarios in this study for both joint space and task space control. (See Fig. 4).

The linear control used to stabilize the robot in the presence of disturbances is given in Eq. (25) and the gains used for the simulations in vector form is given by:

$$k_1 = \begin{bmatrix} 400 & 0 & 0 & 0 & 0 & 0 \\ 0 & 400 & 0 & 0 & 0 & 0 \\ 0 & 0 & 400 & 0 & 0 & 0 \\ 0 & 0 & 0 & 400 & 0 & 0 \\ 0 & 0 & 0 & 0 & 400 & 0 \\ 0 & 0 & 0 & 0 & 0 & 400 \end{bmatrix} \tag{33}$$

and

$$k_0 = \begin{bmatrix} 40 & 0 & 0 & 0 & 0 & 0 \\ 0 & 40 & 0 & 0 & 0 & 0 \\ 0 & 0 & 40 & 0 & 0 & 0 \\ 0 & 0 & 0 & 40 & 0 & 0 \\ 0 & 0 & 0 & 0 & 40 & 0 \\ 0 & 0 & 0 & 0 & 0 & 40 \end{bmatrix}. \tag{34}$$

The same gains were used for the PD control for comparison purposes.

4.1. Trajectory generation in joint space

The problem of generating trajectories involves parameterizing the flat outputs and their derivatives over instants in time t_1 to t_2. The flatness property enables easy mapping between the trajectories in the nonlinear system and that of the flatness space. Terminal conditions for the flat outputs and their derivatives are made equivalent to the initial and final states of the complex manipulator dynamics over a specified period of time. (See Fig. 5.)

Using the boundary conditions at $t_1 = 0$ and $t_2 = 5$ s for the 6 flat outputs and their derivatives, the terminal conditions

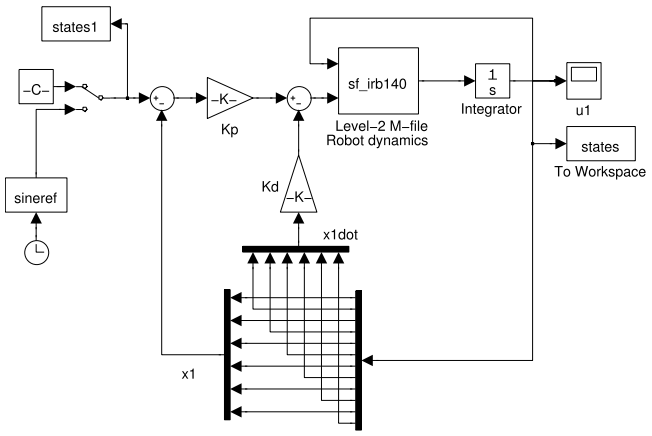

Fig. 5. PID control diagram in simulink.

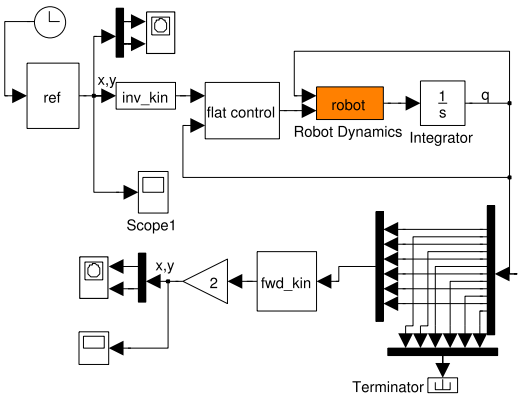

Fig. 6. Task space control in cartesian space.

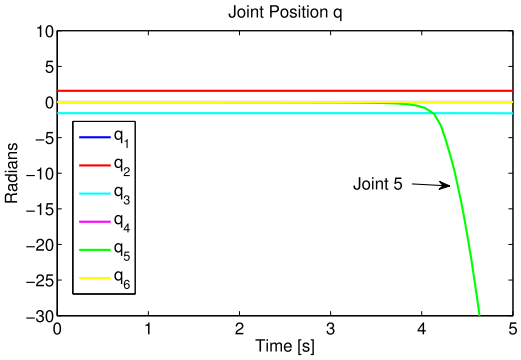

Fig. 7. Open loop with gravity compensation.

for the 12 states are also obtained. A 5th degree polynomial is used to determine the coefficients of the trajectories. The boundary conditions for these trajectories are set bearing in mind the joint limits of the robot model.

$$x_1^*(\tau) = \alpha_0 + \alpha_1\tau + \alpha_2\tau^2 + \alpha_3\tau^3 + \alpha_4\tau^4 + \alpha_5\tau^5 \tag{35}$$

where

$$\tau = \frac{t - t_1}{t_2 - t_1}. \tag{36}$$

Steering the robot about the individual joints, a reference was defined for the joint coordinate and the following were obtained for reference trajectories: For a reference coordinate from origin $(0, 0, 0, 0, 0, 0)$ to $(-pi/2, 0, pi/2, -pi, pi/3, pi)$, we obtained

$$\begin{aligned}
x_{11}^* &= -0.1257\tau^3 + 0.0377\tau^4 - 0.0030\tau^5;\\
x_{12}^* &= 0;\\
x_{13}^* &= 0.1257\tau^3 - 0.0377\tau^4 + 0.0030\tau^5;\\
x_{14}^* &= -0.2513\tau^3 + 0.0754\tau^4 - 0.0060\tau^5;\\
x_{15}^* &= 0.0838\tau^3 - 0.0251\tau^4 + 0.0020\tau^5;\\
x_{16}^* &= 0.2513\tau^3 - 0.0754\tau^4 + 0.0060\tau^5.
\end{aligned} \tag{37}$$

To obtain derivative functions of x_1^*, we differentiate Eq. (37) up to twice.

4.2. Task space control

Using the inverse kinematics equations of Section 2 and the robot jacobian, trajectories were generated for the end effector and converted to joint angles which was applied to the flat controller loop. The output from the control is then converted back to task space trajectories using the forward kinematics of the robot. The block diagram of the control in Simulink is shown in Fig. 6. The robot Jacobian is not shown here since it has already been reported in [27].

5. Simulation, results and discussion

Extensive computer simulations were carried out in Simulink/Matlab environment. Firstly, the open loop dynamics of the Irb140 was simulated under several scenarios. The robot was simulated under gravity and no gravity conditions. The open loop simulations were done using desired torque values that were computed based on the computed dynamics of the robot. Then a feedback flat controller was designed to simulate and stabilize the system in closed loop. The reference trajectories and the complete dynamics of the Irb140 robot are generated using Matlab based S-functions. This enables online computations and updating of its functions with every time step in the simulations.

5.1. Open loop

Simulations were initially done under no gravity and zero torque situation (under specific initial conditions) with the robot being in a stable/rest posture. When the robot is at rest position, i.e. the joint vector initial conditions have the configurations $(0, 0, pi/2, 0, -pi/2, 0)$ and $(0, 0, 0, 0, 0, 0)$ for the velocities respectively. At this position, the gravity terms of the robot are compensated for and hence the robot is at steady state. Moving away from the initial conditions at rest, the robot displays unstable behaviors. An attempt to actuate the robot in open loop results in a chaotic situation as the robot tries to gain balance. An example of such behavior is seen in Fig. 8 (refer to online version for colors). When gravity is added to the dynamics, the situation is more chaotic. This condition persists even when gravity compensation is included in the input torque vector. For example, in Fig. 7, the robot is being driven by a gravity compensating torque. At first, the response is stable and then joint 5 becomes unstable. This behavior shows how highly nonlinear and unpredictable the robot behaves in open loop. For more information on this robot's behavior in open loop, see the work by Herman [36].

Using the differential flatness technique, a feedforward control was designed and used to maneuvre the robot from one boundary point to another. Once these boundaries are set, the robot is easily driven along those boundaries in open loop. However, the initial conditions of the robot in flat space must be the same as that of the real robot. This is the case, in Fig. 9(a), this simulation is done

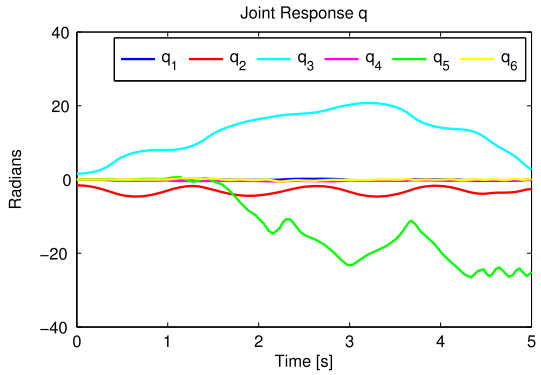

Fig. 8. Open loop without gravity compensation.

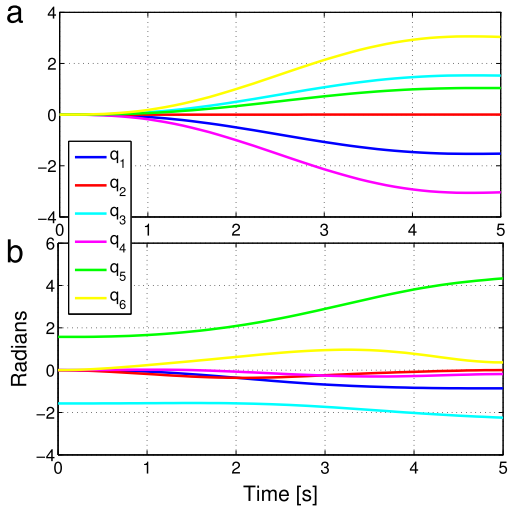

Fig. 9. Open loop using Flatness with gravity compensation.

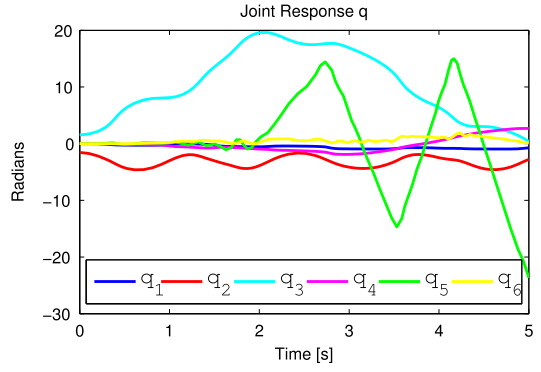

Fig. 10. Open loop using Flatness without gravity compensation.

with gravity compensation. In Fig. 9(b), the initial conditions were varied hence the joint trajectories are not accurate. When the robot is driven in open loop under no gravity compensation situations, unstable response is seen at the various joints. This can be seen in Fig. 10.

5.2. Closed loop trajectory tracking

For tracking the nominal trajectories, the flatness based controller stabilizes the loop and compensates for uncertainties. In this

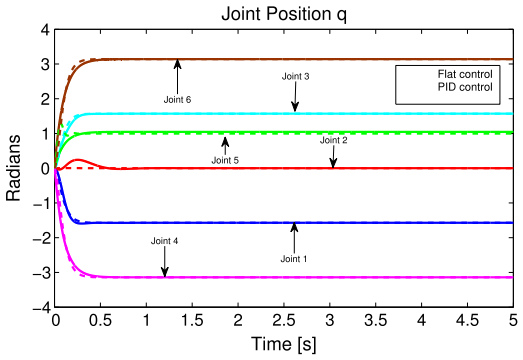

Fig. 11. Set Point Regulation comparing PID and Flat control.

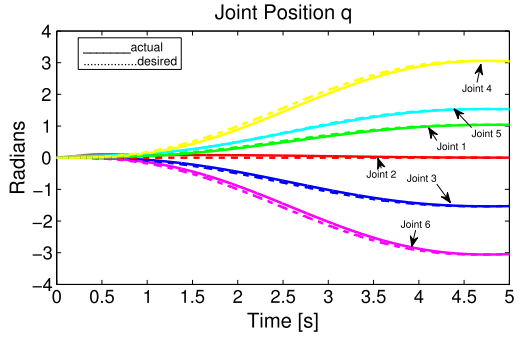

Fig. 12. Trajectory Tracking using PID.

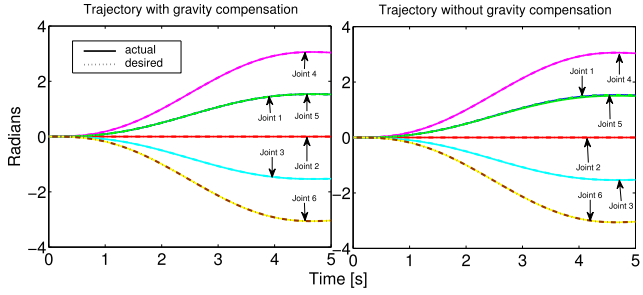

Fig. 13. Trajectory Tracking using Flat control.

study, the nonlinear gravitational effects are taken as disturbances. The function of the controller is to track these trajectories as accurately as possible in the presence of disturbances. The initial conditions of the robot which was observed in open loop is not critical once the feedback flat control is in place. Tracking of the reference trajectories are guaranteed with the flat controller in the presence of disturbances. To compare performance, the robot was regulated for certain setpoints using PD and flat control. The results are shown in Fig. 11. The flat control settles faster than the PID control. In terms of the tracking errors, both PID and flat controller compete favorably. However, the PID control runs into saturation when the controller gains are very high. This is not so with the flat controller.

The flatness generated trajectories were tracked using the PID controller for comparison. With gravity compensation, the tracking response in Fig. 12 shows tracking error of about 0.15 rads. On the other hand, tracking using the flat controller produced accurate results under gravity and no gravity compensation (Fig. 13). The tracking errors using the flatness based control is almost zero.

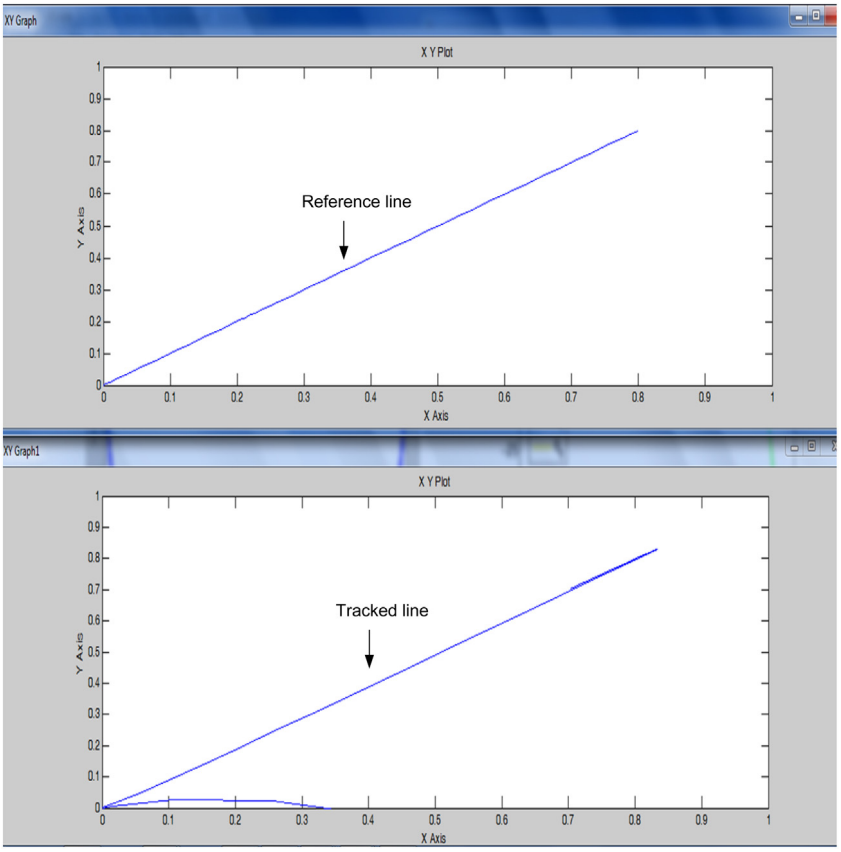

Fig. 14. Tracking a straight line in cartesian space.

The flatness based control is computationally efficient as all simulations in this study were carried out within a few seconds. Compared to the PD control simulation run times in the original study [36], the flatness based control is more than ten times faster.

5.3. Task space trajectory tracking

The results given in the sections above show the effectiveness of the proposed control for trajectory tracking in joint space. Since the manipulator tasks are defined in the operational space, few trajectories planned in cartesian space are used to show the results. Using the set up in SIMULINK of Fig. 6, we first performed simulations using a simple straight line and then an arc.

The robot was able to trace the straight line as shown in Fig. 14 and an arc in Fig. 15.

6. Conclusions

This article has described the flatness based tracking control of a 6-DOF industrial robot manipulator with full dynamics in joint and task space. The flatness based approach to trajectory control offers a fast alternative to PID control for such high dimensional robots. Compared to PID, the gains of the flat controller can be increased to reduce tracking errors without any fear of saturation. Having determined the flat output of the robot, trajectory control was achieved with reasonable accuracy within seconds as compared to when using the PID control. Therefore the proposed control is fast and gives accurate trajectory tracking. This makes it well suited for online practical implementation. In future work, more complex trajectories will be developed and tested with the flatness based controller.

Appendix

A.1. Robot kinematics equations

The expressions for q_2 and q_3 given q_1 are obtained from Fig. 16:

$$q_2 = atan(P_{zw} - d_1, \sqrt{(p_{xw} + a_1 cos(q_1^2)) + (p_{yw} + a_1 sin(q_1))^2}) \\ -atan(d_4 sin(q_3), a_2 + d_4 cos(q_3)) \quad (38)$$

using the law of cosines which is $c^2 = a^2 + b^2 - 2abcosC$, it is easy to see that

$$cos(q_3) = \frac{L^2 + S^2 - a_2^2 - d_4^2}{2a_2 d_4} \quad (39)$$

then

$$q_3 = atan(\pm\sqrt{1 - D^2}, D). \quad (40)$$

Putting:

$$D = cos(q_3) = \\ \frac{(P_{xw} + a_1 cos(q_1))^2 + (p_{yw} + a_1 sin(q_1))^2 + (p_{zw} - d_1)^2 - a_2^2 - d_4^2}{2a_2 d_4} \quad (41)$$

then

$$q_3 = atan(\pm\sqrt{1 - D^2}, D) \quad (42)$$

$$q_4 = atan(s_1 r_{13} - c_1 r_{23}, c_1 c_{23} r_{13} + s_1 c_{23} r_{23} + s_{23} r_{33}) \quad (43)$$

$$q_5 = atan(c_1 c_{23} c_4 + s_1 s_4) r_{13} + (s_1 c_{23} c_4 - c_1 s_4) r_{23} + s_{23} c_4 r_{33}, \\ (c_1 s_{23}) r_{13} + (s_1 s_{23}) r_{23} - c_{23} r_{33} \quad (44)$$

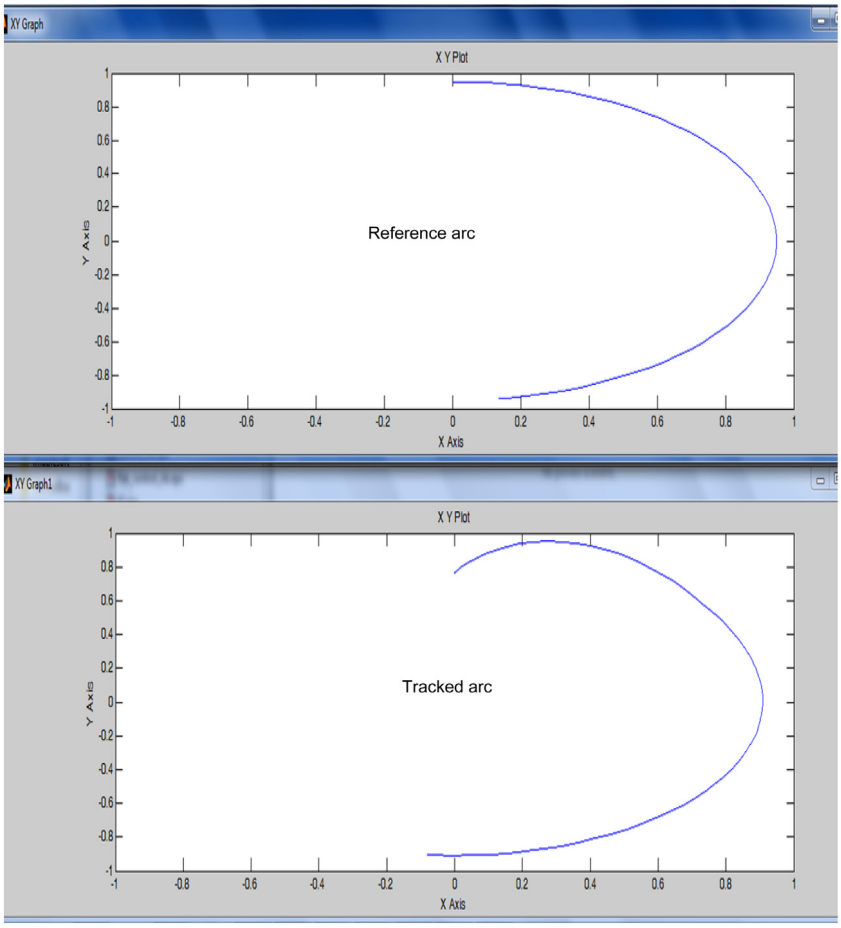

Fig. 15. Tracking an arc in Cartesian space.

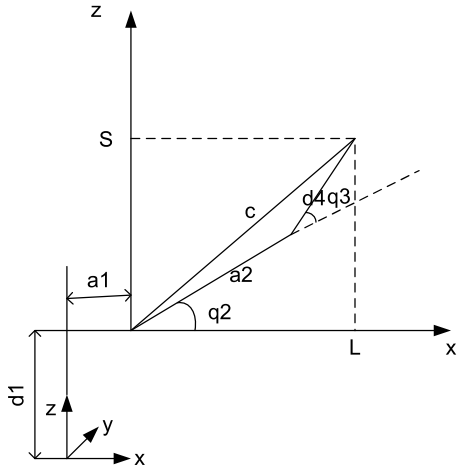

Fig. 16. Projection onto the plane formed by links 2 and 3.

$$q_6 = atan(-c_1c_{23}s_4 + s_1c_4)r_{11} - (s_1c_{23}s_4 - c_1c_4)r_{21} - s_{23}s_4r_{31},\\ (-c_1s_{23}s_4 + s_1c_4)r_{12} - (s_1c_{23}s_4 + c_1c_4)r_{22} - s_{23}s_4r_{32}. \tag{45}$$

A.2. Robot dynamic equations

Some of the dynamic equations used for the study are defined in this section. Because of large size of the equations from maple computations, only part is shown here.

$M(q)$ is defined as

$$\begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} & m_{15} & m_{16} \\ m_{21} & m_{22} & m_{23} & m_{24} & m_{25} & m_{26} \\ m_{31} & m_{32} & m_{33} & m_{34} & m_{35} & m_{36} \\ m_{41} & m_{42} & m_{43} & m_{44} & m_{45} & m_{46} \\ m_{51} & m_{52} & m_{53} & m_{54} & m_{55} & m_{56} \\ m_{61} & m_{62} & m_{63} & m_{64} & m_{65} & m_{66} \end{bmatrix} \tag{A.1}$$

$$\begin{aligned} m_1 &= \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} & m_{15} & m_{16} \end{bmatrix} \\ m_2 &= \begin{bmatrix} m_{21} & m_{22} & m_{23} & m_{24} & m_{25} & m_{26} \end{bmatrix} \\ m_3 &= \begin{bmatrix} m_{31} & m_{32} & m_{33} & m_{34} & m_{35} & m_{36} \end{bmatrix} \\ m_4 &= \begin{bmatrix} m_{41} & m_{42} & m_{43} & m_{44} & m_{45} & m_{46} \end{bmatrix} \\ m_5 &= \begin{bmatrix} m_{51} & m_{52} & m_{53} & m_{54} & m_{55} & m_{56} \end{bmatrix} \\ m_6 &= \begin{bmatrix} m_{61} & m_{62} & m_{63} & m_{64} & m_{65} & m_{66} \end{bmatrix}. \end{aligned} \tag{A.2}$$

Putting

$$\begin{aligned} x_1 &= q_1 \\ x_2 &= q_2 \\ x_3 &= q_3 \\ x_4 &= q_4 \\ x_5 &= q_5 \\ x_6 &= q_6 \\ x_7 &= \dot{q}_1 \\ x_8 &= \dot{q}_2 \\ x_9 &= \dot{q}_3 \\ x_{10} &= \dot{q}_4 \\ x_{11} &= \dot{q}_5 \\ x_{12} &= \dot{q}_6 \end{aligned} \tag{A.3}$$

from Eq. (11)

$$f(x) = \begin{bmatrix} x_7 \\ x_8 \\ x_9 \\ x_{10} \\ x_{11} \\ x_{12} \\ M(q)^{-1}(-C(q,\dot{q}) - G(q)) \end{bmatrix} \tag{A.4}$$

$$g(x) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ m_{11} & m_{12} & m_{13} & m_{14} & m_{15} & m_{16} \\ m_{21} & m_{22} & m_{23} & m_{24} & m_{25} & m_{26} \\ m_{31} & m_{32} & m_{33} & m_{34} & m_{35} & m_{36} \\ m_{41} & m_{42} & m_{43} & m_{44} & m_{45} & m_{46} \\ m_{51} & m_{52} & m_{53} & m_{54} & m_{55} & m_{56} \\ m_{61} & m_{62} & m_{63} & m_{64} & m_{65} & m_{66} \end{bmatrix} \tag{A.5}$$

where:

$m_{11} = cos((q5+2pi-2q4-2q3-2q2))0.216840434497100888e-18+cos((-2q2+pi))0.224527199999999994e1+cos((2q4+2q3+2q2+q5))0.216840434497100888e-18+cos((-2q4+2q3+2q2+q5))0.216840434497100888e-18+sin(-q2+pi/0.2e1-q3)0.333200000000000052e0+0.6800000000e-4cos((2q5+3pi+q4-2q3-2q2))-0.1700000000e-4cos((2q5+pi+2q3+2q2-2q4))+cos((q5+q4+2q3+2q2))0.275499999999999964e-2+cos((q5+2pi-q4-2q3-2q2))(-0.275499999999999964e-2)+sin((q5+pi-q4+q3))0.260999999999999991e-2+cos((q5+2q2+2q3-q4))0.275499999999999964e-2+cos((pi+2q4+q5))(-0.433680868994201774e-18)-0.6800000000e-4cos((-q4+2q5+pi+2q3+2q2))+sin((q5+2pi-q4-q3-2q2))0.260999999999999991e-2+sin((q5+q4+q3+2q2))0.260999999999999991e-2+cos((q5+2pi-2q3-2q2))0.551000000000000101e-2+sin((q5+pi+q3))0.521999999999999982e-2+sin((q5+pi-q3))(-0.521999999999999982e-2)+cos(-q2+pi/0.2e1)0.192948000000000008e1+cos((-2q2+pi+2q4))0.138777878078144570e-16+sin(q3)(-0.856800000000000006e0)-0.1700000000e-4cos((2q5+3pi-2q3-2q2+2q4))+0.3400000000e-4cos((2q4+2pi+2q5))+cos((2q4-2q3-2q2+pi))0.340000000000062474e-4+cos((q5+2pi+2q4-2q3-2q2))0.216840434497100888e-18+sin((-q3-2q2+pi))0.856800000000000006e0-0.6800000000e-4cos((q4+2q5+pi+2q3+2q2))+cos((q5+pi))(-0.110200000000000020e-1)+cos((q5+2q2+2q3))0.551000000000000101e-2+sin(q5+0.3e1/0.2e1pi-q3-q2)(-0.203000000000000012e-2)+sin((q5+pi+q4-q3))0.260999999999999991e-2+sin((q5+pi+q4+q3))0.260999999999999991e-2+sin((q5+pi-q4-q3))0.260999999999999991e-2+cos((pi-2q4+q5))(-0.433680868994201774e-18)-0.1700000000e-4cos((2q5+3pi-2q4-2q3-2q2))+0.6800000000e-4cos((2q5+3pi-q4-2q3-2q2))+sin((q5+2pi+q4-q3-2q2))0.260999999999999991e-2+sin((q5-q4+q3+2q2))0.260999999999999991e-2-0.6800000000e-4cos((2pi+2q5))+cos((-2q2+pi-2q3))(-0.499655500000000030e0)+0.3400000000e-4cos((-2q4+2pi+2q5))-0.1020000000e-3cos((2q5+3pi-2q3-2q2))-0.1020000000e-3cos((2q5+pi+2q3+2q2))+sin(q5+pi/0.2e1+q3+q2)0.203000000000000012e-2+cos((q5+2pi-2q2-2q3+q4))(-0.275499999999999964e-2)-0.1700000000e-4cos((2q5+pi+2q4+2q3+2q2))+cos((2q4))(-0.679999999999847394e-4)+cos((-2q4-2q3-2q2+pi))0.340000000000062474e-4+sin((q5+2pi-q3-2q2))(-0.521999999999999982e-2)+sin((q5+q3+2q2))0.521999999999999982e-2+0.374662799999999940e1+sin(q5+pi/0.2e1+q3+q2+q4)0.101500000000000006e-2+sin(q5+0.3e1/0.2e1pi-q3-q2+q4)0.101500000000000006e-2+sin(q5+pi/0.2e1+q3+q2-q4)0.101500000000000006e-2+sin(q5+0.3e1/0.2e1pi-q3-q2-q4)0.101500000000000006e-2+cos((-2q2+pi-2q4))0.138777878078144570e-16.$

References

[1] R. Tedrake, Underactuated Robotics: Learning, Planning, and Control for Efficient and Agile Machines Course Notes for MIT 6.832, Working draft edition, 2009.

[2] N. Kumar, V. Panwar, J.-H. Borm, J. Chai, Enhancing precision performance of trajectory tracking controller for robot manipulators using RBFNN and adaptive bound, Appl. Math. Comput. 231 (2014) 320–328.

[3] C.Y. Lai, Improving the transient performance in robotics force control using nonlinear damping, in: Advanced Intelligent Mechatronics (AIM), 2014 IEEE/ASME International Conference on, 2014, pp. 892–897.

[4] A. Lazinica, H. Kawai, Robot Manipulators, New Achievements, In-Tech, Vukovar, Croatia, 2010.

[5] G.G. Rigatos, Model-based and model-free control of flexible-link robots: a comparison between representative methods, Appl. Math. Model. 33 (10) (2009) 3906–3925.

[6] H. Abdellatif, B. Heimann, Advanced model-based control of a 6-DOF hexapod robot: A case study, IEEE/ASME Trans. Mechatronics 15 (2) (2010) 269–279.

[7] C. Tang, P. Miller, V. Krovi, J. Ryu, S. Agrawal, Differential-flatness-based planning and control of a wheeled mobile manipulator–theory and experiment, IEEE/ASME Trans. Mechatronics (99) (2010) 1–6.

[8] J.-C. Ryu, S.K. Agrawal, Planning and control of under-actuated mobile manipulators using differential flatness, Auton. Robots 29 (1) (2010) 35–52.

[9] J. Ryu, S. Agrawal, Differential flatness-based robust control of mobile robots in the presence of slip, Int. J. Robot. Res. 30 (4) (2011) 463.

[10] D.B. Vicente, Modeling and Balancing of Spherical Pendulum using a Parallel Kinematic Manipulator, 2007.

[11] S.K. Agrawal, V. Sangwan, Differentially flat designs of underactuated open-chain planar robots, IEEE Trans. Robot. 24 (6) (2008) 1445–1451.

[12] J.J. Craig, Introduction To Robotics: Mechanics and Control, Prentice Hall, Englewood Cliffs, NJ, 2005.

[13] M. Fliess, J. Lvine, P. Martin, F. Ollivier, P. Rouchon, Controlling nonlinear systems by flatness, Syst. Control Twenty-First Century (1997) 137–154.

[14] M. Fliess, J. Lvine, P. Martin, P. Rouchon, 1995.

[15] P. Rouchon, M. Fliess, J. Lvine, P. Martin, Flatness, Motion Planning and Trailer Systems, vol. 3, IEEE, 1993.

[16] H. Sira-Ramirez, S.K. Agrawal, Differentially Flat Systems, Vol. 17, CRC, 2004.

[17] E.V. Diaz, J. Slama, M. Dutra, O. Lengerke, M.M. Tavera, Trajectory tracking for robot manipulators using differential flatness, Ingr. e Investig. 31 (2) (2011) 84–90.

[18] J. Levine, Analysis and Control of Nonlinear Systems: A Flatness-Based Approach, Springer, 2009.

[19] J. Coulaud, G. Campion, Optimal trajectory tracking for differentially flat systems with singularities, in: Control and Automation. ICCA 2007. IEEE International Conference on, IEEE, 2007, pp. 1960–1965.

[20] R. Rao, V. Kumar, C. Taylor, Visual servoing of a UGV from a UAV using differential flatness, in: Intelligent Robots and Systems (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on, vol. 1, IEEE, 2003, pp. 743–748.

[21] E. Markus, J. Agee, A. Jimoh, N. Tlale, B. Zafer, Flatness based control of a 2 DOF single link flexible joint manipulator, in: 2nd International Conference on Simulation and Modeling Methodologies, Technologies and Applications, SciTePress, 2012, pp. 437–442.

[22] F. Bullich, S. Agrawal, V. Sangwan, Differential Flatness of a Class of N-DOF Planar Manipulators Driven By 1 Or 2 Actuators, 2010.

[23] J. Franch, S.K. Agrawal, Design of differentially flat planar space robots and their planning and control, Internat. J. Control 81 (3) (2008) 407–416.

[24] S.K. Agrawal, K. Pathak, J. Franch, R. Lampariello, G. Hirzinger, A differentially flat open-chain space robot with arbitrarily oriented joint axes and two momentum wheels at the base, IEEE Trans. Autom. Control 54 (9) (2009) 2185–2191.

[25] E.D. Markus, J.T. Agee, A.A. Jimoh, Trajectory control of a two-link robot manipulator in the presence of gravity and friction, in: AFRICON, IEEE, 2013, pp. 1–5.

[26] E. Markus, J. Agee, A. Jimoh, Differentially flat trajectory control of a 6DOF industrial robot manipulator, in: International Association for Science and Technology (IASTED) Control and Application Conference, Iasted, 2013.

[27] T.J. Carter, The Modeling of a Six Degree-of-Freedom Industrial Robot for the Purpose of Efficient Path Planning, (Ph.D. dissertation), 2009.

[28] J.M. Hollerbach, A recursive Lagrangian formulation of manipulator dynamics and a comparative study of dynamics formulation complexity, IEEE Trans. Syst. Man Cybern. 10 (11) (1980) 730–736.
[29] ABB, ABB Robotics Product Manual IRB 140, 2004.
[30] J. Levine, On necessary and sufficient conditions for differential flatness, Arxiv preprint , 2006.
[31] M. Fliess, J. Lvine, P. Martin, P. Rouchon, A Lie–Backlund approach to equivalence and flatness of nonlinear systems, IEEE Trans. Autom. Control 44 (5) (1999) 922–937.
[32] J.-J.E. Slotine, W. Li, Applied Nonlinear Control, Prentice-Hall, Englewood Cliffs, NJ, 1991.
[33] G. Rigatos, P. Siano, P. Wira, V. Loia, A PEM fuel cells control approach based on differential flatness theory, Intell. Ind. Syst. (2016) 1–11.
[34] A. Isidori, Nonlinear Control Systems II, Vol. 2, Springer Verlag, 1999.
[35] M.J. Van Nieuwstadt, Trajectory Generation for Nonlinear Control Systems, (Ph.D. dissertation), 1996.
[36] H. Hoifodt, Dynamic Modeling and Simulation of Robot Manipulators: The Newton–Euler Formulation, (Ph.D. dissertation), 2011.

Elisha D. Markus received his B.Eng and M.Eng. Degrees from Abubakar Tafawa Balewa University, Nigeria in 1997 and 2001 respectively. He worked in the Telecommunications industry for about 10 years before joining the university as a lecturer. He is currently a doctoral student at Tshwane University of Technology in South Africa and also a lecturer at Central University of Technology in the Free State South Africa. His research interests include nonlinear control, robotics, Power systems, Differential Flatness, Telecommunications and artificial intelligence.

John T. Agee was born in 1964. He obtained a B.Eng. (Hons), 2nd Class Upper division in Electrical and Electronics Engineering in 1989, an M.Eng (Electronics) degree in 1992 and a Ph.D. in Control Systems Engineering in 2001; all from the Abubakar Tafawa Balewa University, Bauchi, Nigeria. He started his academic career in 1990 as a Graduate Assistant. He became a Senior Lecturer in 2001. His research interests include nonlinear control with applications in power systems. He is also researching in artificial intelligence and its application in smart drive control.

Adisa A. Jimoh received B.Eng. degree in 1977 and M.Eng. degree in 1980, both from Ahmadu Bello University (ABU) Zaria, Nigeria and a Ph.D. degree from McMaster University, Hamilton, Canada in 1986. He was with Ahmadu Bello University up till 1992, when he moved to the Research & Development unit of the National Electric Power Authority (NEPA) Nigeria. Adisa was with NEPA up till 1996 when he relocated to the University of Durban-Westville, Durban, South Africa where he taught courses and carried out research in high performance energy efficient electric machines and power systems, and power electronics. In 2001 he joined Tshwane University of Technology, Pretoria, where, as a full professor, he leads and coordinate the research and doctorate programs of the Graduate School of Electrical and Electronic Engineering. He is a registered engineer in South Africa. His research interests are in the field of Electric Machines, Drives and Power Electronics application in Power Systems.