DE GRUYTER OPEN

Journal of MECHANICAL ENGINEERING – Strojnícky časopis,
VOL 67 (2017), NO 2, 101 - 108

STU SjF

PROPOSED ROBOT SCHEME WITH 5 DOF AND DYNAMIC MODELLING USING MAPLE SOFTWARE

SHALA Ahmet[1], BRUÇI Mirlind[1*]

[1] Faculty of Mechanical Engineering, University of Prishtina, Bregu i Diellit, p.n. 10000 Prishtina, Kosovo
** Corresponding author e-mail: mirlind.bruci@uni-pr.edu*

Abstract: In this paper is represented Dynamical Modelling of robots which is commonly first important step of Modelling, Analysis and Control of robotic systems. This paper is focused on using Denavit-Hartenberg (DH) convention for kinematics and Newton-Euler Formulations for dynamic modelling of 5 DoF - Degree of Freedom of 3D robot. The process of deriving of dynamical model is done using Software Maple. Derived Dynamical Model of 5 DoF robot is converted for Matlab use for future analysis, control and simulations.

KEYWORDS: Modelling, dynamics, robot, analysis software's

1 Introduction

Dynamics is a huge field of study devoted to studying the forces required to cause motion. The dynamic motion of the manipulator arm in a robotic system is produced by the torques generated by the actuators [1, 2].

The relationship between the input torques and the time rates of change of the robot arm components configurations, represent the dynamic modelling of the robotic system which is concerned with the derivation of the equations of motion of the manipulator as a function of the forces and moments acting on. So, the dynamic modelling of a robot manipulator consists of finding the mapping between the forces exerted on the structures and the joint positions, velocities and accelerations. A good model has to satisfy two conflicting objectives [4, 5].

A robot manipulator is basically a positioning device. To control the position we must know the dynamic properties of the manipulator in order to know how much force to exert on it to cause it to move: too little force and the manipulator is slow to react; too much force and the arm may crash into objects or oscillate about its desired position.

Deriving the dynamic equations of motion for robots is not a simple task due to the large number of degrees of freedom and nonlinearities present in the system [2, 3,].

This part is concerned with the development of the dynamic model for 5 DoF robot and their kinematics and dynamics equations.

Proposed robot scheme is under construction at Faculty of Mechanical Engineering, the theoretical and practical study in future will be oriented on control and study of existence of propulsion effect.

2 Structure and coordinate systems of 5 DoF robot

Based on structure of 5 DoF robot (Figure 1), is created Table 1 of Denavit-Hartenberg parameters for 5 DoF robot.

DOI: 10.1515/scjme-2017-0023, Print ISSN 0039-2472, On-line ISSN 2450-5471

Tab. 1 Denavit-Hartenberg parameters for 5 DoF robot.

Link #	a_i	α_i	d_i	θ_i
1	0	0	d_1	$q_1^* + \frac{\pi}{2}$
2	0	$\frac{\pi}{2}$	d_2^*	0
3	0	$\frac{\pi}{2}$	0	$q_3^* + \frac{\pi}{2}$
4	0	$\frac{\pi}{2}$	$L_3 + L_4$	$q_4^* + \frac{\pi}{2}$
5	0	0	d_5^*	0

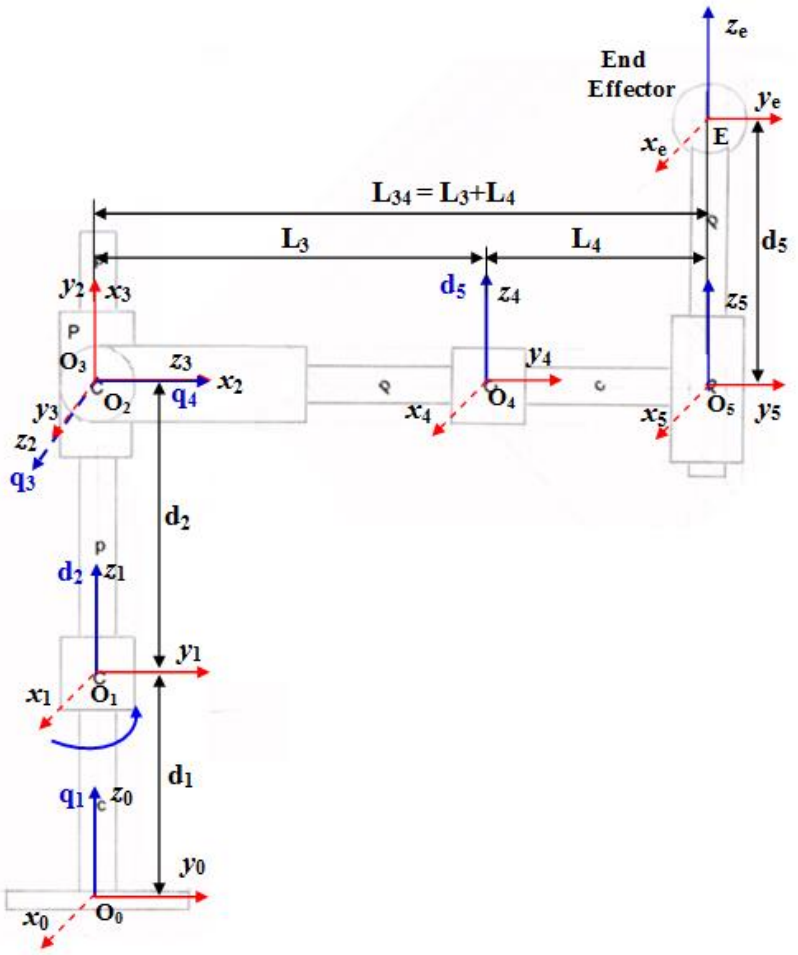

Fig. 1 Symbolic representation - Axes rotations for Denavit-Hartenberg parameters

Orthogonal rotation matrix R_i which transforms a vector in the i-th coordinate frame to a coordinate frame which is parallel to the (i-1)-th coordinate frame is first 3x3 sub-matrices of A_i:

$$R_i = \begin{bmatrix} \cos(\theta_i) & -\cos(\alpha_i)\cdot\sin(\theta_i) & \sin(\alpha_i)\cdot\sin(\theta_i) \\ \sin(\theta_i) & \cos(\alpha_i)\cdot\cos(\theta_i) & -\sin(\alpha_i)\cdot\cos(\theta_i) \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) \end{bmatrix}$$

for i=1,2, … , N, where $R_{N+1} = E = diag(1)$.

3 Dynamic Equations – Newton-Euler Formulation

Dynamics of robot is the study of motion with regard to forces (the study of the relationship between forces/torques and motion). A dynamic analysis of a manipulator is useful for the following purposes:

1- It determines the joint forces and torques required to produce specified end-effector motions (the direct dynamic problem).

2- It produces a mathematical model which simulates the motion of the manipulator under various loading conditions (the inverse dynamic problem) and/or control schemes.

3- It provides a dynamic model for use in the control of the actual manipulator.

Dynamic modelling of mechanical structures can be a complex problem. In robotics, more specifically, in manipulators, there are two methodologies used for dynamic modelling: The Newton-Euler formulation and Lagrange Equations. We have decided for Newton-Euler formulation because of known that this method is compactible for use on computer.

Newton-Euler formulation

The Newton-Euler formulation [1] shown in equations (1)-(9) computes the inverse dynamics (i.e., joint torques/forces from joint positions, velocities, and accelerations) bases on two sets of recursions: the *forward* and *backward* recursions. The forward recursions (1)-(3) transform the kinematics variables from the base to the end-effector. The initial conditions (for i=0) assume that the manipulator is at rest in the gravitational field. The backward recursions (4)-(9) transform the forces and moments from the end-effector to the base, and culminate with the calculation of the joint torques/forces.

Angular velocity of the i-th coordinate frame:

$$\omega_{i+1} = \begin{cases} R_{i+1}^T \cdot \left[\omega_i + z_0 \cdot \dot{\theta}_{i+1}\right] & \text{if joint is rotational} \\ R_{i+1}^T \cdot \omega_i & \text{if joint is translational} \end{cases} \tag{1}$$

where: $z_0 = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$

Angular acceleration of the i-th coordinate frame

$$\dot{\omega}_{i+1} = \begin{cases} R_{i+1}^T \cdot \left[\dot{\omega}_i + z_0 \cdot \ddot{\theta}_{i+1} + \omega_i \times (z_0 \cdot \dot{\theta}_{i+1})\right] & \text{if joint is rotational} \\ R_{i+1}^T \cdot \dot{\omega}_i & \text{if joint is translational} \end{cases} \tag{2}$$

Linear acceleration of the i-th coordinate frame

$$\dot{v}_{i+1} = \begin{cases} R_{i+1}^T \cdot \dot{v}_i + \dot{\omega}_{i+1} \times p_{i+1} + \omega_{i+1} \times (\omega_{i+1} \times p_{i+1}) \\ \text{if joint is rotational} \\ R_{i+1}^T \cdot \left[\dot{v}_i + z_0 \cdot \ddot{d}_{i+1} + 2\omega_i \times (z_0 \cdot \dot{d}_{i+1})\right] + \dot{\omega}_{i+1} \times p_{i+1} + \omega_{i+1} \times (\omega_{i+1} \times p_{i+1}) \\ \text{if joint is translational} \end{cases} \tag{3}$$

where $p_i = \begin{bmatrix} a_i & d_i \cdot \sin(\alpha_i) & d_i \cdot \sin(\alpha_i) \end{bmatrix}^T$ is position of the i-th coordinate frame with respect to the $(i-1)$-th coordinate frame.

Initial conditions:

$$\omega_0 = 0, \quad \dot{\omega}_0 = 0, \quad v_0 = 0$$

Gravitational acceleration:

$$\dot{v}_0 = \begin{bmatrix} g_x & g_y & g_z \end{bmatrix}^T .$$

Linear acceleration of the centre-of-mass of link *i*

$$a_i = \dot{\omega}_i \times s_i + \omega_i \times (\omega_i \times s_i) + \dot{v}_i \tag{4}$$

where: s_i is position of centre-of-mass of link *i*

Net force exerted on link *i:*

$$F_i = m_i \cdot a_i \tag{5}$$

Net moment exerted on link *i*

$$N_i = I_i \dot{\omega}_i + \omega_i \times (I_i \cdot \omega_i) \tag{6}$$

where

$$I_i = \begin{bmatrix} I_{ixx} & 0 & 0 \\ 0 & I_{iyy} & 0 \\ 0 & 0 & I_{izz} \end{bmatrix}$$

I_i is moment of inertia tensor of link *i* about the centre-of-mass of link *i* (parallel to the *i*-th coordinate frame), with only principal inertias I_{ixx}, I_{iyy} and I_{izz}. Because of symmetry of link frames, cross-inertias can be used zero.

Force exerted on link *i* by link *i* – 1:

$$f_i = R_{i+1}^T \cdot f_{i+1} + F_i \tag{7}$$

Moment exerted on link ***i*** by link *i* – 1

$$n_i = R_{i+1}^T \cdot n_{i+1} + p_i \times f_i + N_i + s_i \times F_i \tag{8}$$

Joint torque/force at joint ***i:***

$$\tau_i = \begin{cases} n_i^T \cdot (R_{i+1}^T \cdot z_0) & \text{if joint is rotational} \\ f_i^T \cdot (R_{i+1}^T \cdot z_0) & \text{if joint is translational} \end{cases} \tag{9}$$

Modelling of **5 DoF robot** is done using **Maple software**, equations are converted for Matlab simulations use.

4 Listing of derived Dynamical Model for 5 DoF using Maple software

```
> # Start of Maple file, Dynamical model of 5 DoF robot using Maple
> # Symbolic representation of Robot - Axes rotations for Denavit-Hartenberg parameters
> restart :
> with(LinearAlgebra) :
> # Joint variables (degrees of freedom)
> q := Vector([[q1], [d2], [q3], [q4], [d5]]) :
> dq := Vector([[dq1], [dd2], [dq3], [dq4], [dd5]]) :
> ddq := Vector([[ddq1], [ddd2], [ddq3], [ddq4], [ddd5]]) :
> # Link length vectors and vectors to the centres of mass
> s1 := Vector([[s1x], [s1y], [s1z]]) :
> s2 := Vector([[s2x], [s2y], [s3z]]) :
```

```
> s3 := Vector([[s3x], [s3y], [s3z]]) :
> s4 := Vector([[s4x], [s4y], [s4z]]) :
> s5 := Vector([[s5x], [s5y], [s5z]]) :
> # Rotation matrices
> R1 := Matrix([[-sin(q[1]), -cos(q[1]), 0], [cos(q[1]), -sin(q[1]), 0], [0, 0, 1]]) :
  R2 := Matrix([[1, 0, 0], [0, 0, -1], [0, 1, 0]]) :
  R3 := Matrix([[-sin(q[3]), 0, cos(q[3])], [cos(q[3]), 0, sin(q[3])], [0, 1, 0]]) :
> R4 := Matrix([[-sin(q[4]), 0, cos(q[4])], [cos(q[4]), 0, sin(q[4])], [0, 1, 0]]) :
  R5 := Matrix([[1, 0, 0], [0, 1, 0], [0, 0, 1]]) :
> # Initial conditions
> z0 := Vector([[0], [0], [1]]) :
> ω0 := Vector([[0], [0], [0]]) :
> dω0 := Vector([[0], [0], [0]]) :
> v0 := Vector([0], [0], [0]) :
> dv0 := Vector([[0], [0], [g]]) :
> # Forward recursions (i=1, 2, 3, 4, 5)
> # Angular velocity of the i-th coordinate frame
> ω1 := combine(MatrixVectorMultiply(Transpose(R1), (ω0 + z0·dq[1]))) :
> ω2 := combine(MatrixVectorMultiply(Transpose(R2), ω1)) :
> ω3 := combine(MatrixVectorMultiply(Transpose(R3), (ω2 + z0·dq[3]))) :
> ω4 := combine(MatrixVectorMultiply(Transpose(R4), (ω3 + z0·dq[4]))) :
> ω5 := combine(MatrixVectorMultiply(Transpose(R5), ω4)) :
> # Angular acceleration of the i-th coordinate frame
> dω1 := combine(MatrixVectorMultiply(Transpose(R1), (dω0 + z0·ddq[1]
      + CrossProduct(ω0, z0·dq[1])))) :
> dω2 := combine(MatrixVectorMultiply(Transpose(R2), dω1)) :
> dω3 := combine(MatrixVectorMultiply(Transpose(R3), (dω2 + z0·ddq[3]
      + CrossProduct(ω2, z0·dq[3])))) :
> dω4 := combine(MatrixVectorMultiply(Transpose(R4), (dω3 + z0·ddq[4]
      + CrossProduct(ω3, z0·dq[4])))) :
> dω5 := combine(MatrixVectorMultiply(Transpose(R5), dω4)) :
> # Position of the i-th coordinate frame with respect to the (i-1)-th coordinate frame
> p1 := Vector([[0], [0], [d1]]) :
> p2 := Vector([[0], [d2], [0]]) :
> p3 := Vector([[0], [0], [0]]) :
> p4 := Vector([[0], [L34], [0]]) :
> p5 := Vector([[0], [0], [d5]]) :
> # Linear acceleration of the i-th coordinate frame
>dv1 := combine(MatrixVectorMultiply(Transpose(R1), dv0) + CrossProduct(dω1, p1) + CrossProduct(ω1, CrossProduct(ω1, p1))) :
> dv2 := combine(MatrixVectorMultiply(Transpose(R2), dv1 + z0·ddq[2] + CrossProduct(2·ω1, z0·dq[2])) + CrossProduct(dω2, p2)
      + CrossProduct(ω2, CrossProduct(ω2, p2))) :
> dv3 := combine(MatrixVectorMultiply(Transpose(R3), dv2) + CrossProduct(dω3, p3) + CrossProduct(ω3, CrossProduct(ω3, p3))) :
> dv4 := combine(MatrixVectorMultiply(Transpose(R4), dv3) + CrossProduct(dω4, p4) + CrossProduct(ω4, CrossProduct(ω4, p4))) :
> dv5 := combine(MatrixVectorMultiply(Transpose(R5), dv4 + z0·ddq[5] + CrossProduct(2·ω4, z0·dq[5])) + CrossProduct(dω5, p5)
      + CrossProduct(ω5, CrossProduct(ω5, p5))) :
> # Linear acceleration of the centre-of-mass of link i
> da1 := combine(CrossProduct(dω1, s1) + CrossProduct(ω1, CrossProduct(ω1, s1)) + dv1) :
> da2 := combine(CrossProduct(dω2, s2) + CrossProduct(ω2, CrossProduct(ω2, s2)) + dv2) :
> da3 := combine(CrossProduct(dω3, s3) + CrossProduct(ω3, CrossProduct(ω3, s3)) + dv3) :
> da4 := combine(CrossProduct(dω4, s4) + CrossProduct(ω4, CrossProduct(ω4, s4)) + dv4) :
> da5 := combine(CrossProduct(dω5, s5) + CrossProduct(ω5, CrossProduct(ω5, s5)) + dv5) :
> # Moment of inertia tensor of link i about the centre-of-mass of link i (parallel to the i-th coordinate frame),
with only principal inertias Iixx, Iiyy and Iizz. Because of symmetry of link frames, cross-inertias are used zero.
```

$$> I1 := \begin{bmatrix} I1xx & 0 & 0 \\ 0 & I1yy & 0 \\ 0 & 0 & I1zz \end{bmatrix}: \quad I2 := \begin{bmatrix} I2xx & 0 & 0 \\ 0 & I2yy & 0 \\ 0 & 0 & I2zz \end{bmatrix}: \quad I3 := \begin{bmatrix} I3xx & 0 & 0 \\ 0 & I3yy & 0 \\ 0 & 0 & I3zz \end{bmatrix}:$$

$$> I4 := \begin{bmatrix} I4xx & 0 & 0 \\ 0 & I4yy & 0 \\ 0 & 0 & I4zz \end{bmatrix}: \quad I5 := \begin{bmatrix} I5xx & 0 & 0 \\ 0 & I5yy & 0 \\ 0 & 0 & I5zz \end{bmatrix}:$$

> # Net force and moment exerted on link *i*

> $F1 := m1 \cdot da1:\ F2 := m2 \cdot da2:\ F3 := m3 \cdot da3:\ F4 := m4 \cdot da4:\ F5 := m5 \cdot da5:$

> $N1 := combine(MatrixVectorMultiply(I1, d\omega1) + CrossProduct(\omega1, MatrixVectorMultiply(I1, \omega1))):$

> $N2 := combine(MatrixVectorMultiply(I2, d\omega2) + CrossProduct(\omega2, MatrixVectorMultiply(I2, \omega2))):$

> $N3 := combine(MatrixVectorMultiply(I3, d\omega3) + CrossProduct(\omega3, MatrixVectorMultiply(I3, \omega3))):$

> $N4 := combine(MatrixVectorMultiply(I4, d\omega4) + CrossProduct(\omega4, MatrixVectorMultiply(I4, \omega4))):$

> $N5 := combine(MatrixVectorMultiply(I5, d\omega5) + CrossProduct(\omega5, MatrixVectorMultiply(I5, \omega5))):$

> # Backward recursion (*i*=5, 4, 3, 2, 1); Force and moment exerted on link *i* by link *i*-1 by supposing that there are no outside load on end-effector

> # End effector

$$> f6 := \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}: \quad n6 := \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}: \quad R6 := \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}:$$

> # Link #5

> $f5 := MatrixVectorMultiply(R6, f6) + F5:$

> $n5 := MatrixVectorMultiply(R6, n6) + CrossProduct(p5, f5) + N5 + CrossProduct(s5, F5):$

> # Force at joint 5

> $\tau5 := MatrixVectorMultiply(Transpose(f5), MatrixVectorMultiply(Transpose(R5), z0)):$

> # Link #4

> $f4 := MatrixVectorMultiply(R5, f5) + F4:$

> $n4 := MatrixVectorMultiply(R5, n5) + CrossProduct(p4, f4) + N4 + CrossProduct(s4, F4):$

> # Torque at joint 4

> $\tau4 := MatrixVectorMultiply(Transpose(n4), MatrixVectorMultiply(Transpose(R4), z0)):$

> # Link #3

> $f3 := MatrixVectorMultiply(R4, f4) + F3:$

> $n3 := MatrixVectorMultiply(R4, n4) + CrossProduct(p3, f3) + N3 + CrossProduct(s3, F3):$

> # Torque at joint 3

> $\tau3 := MatrixVectorMultiply(Transpose(n3), MatrixVectorMultiply(Transpose(R3), z0)):$

> # Link #2

> $f2 := MatrixVectorMultiply(R3, f3) + F2:$

> $n2 := MatrixVectorMultiply(R3, n3) + CrossProduct(p2, f2) + N2 + CrossProduct(s2, F2):$

> # Force at joint 2

> $\tau2 := MatrixVectorMultiply(Transpose(f2), MatrixVectorMultiply(Transpose(R2), z0)):$

> # Link #1

> $f1 := MatrixVectorMultiply(R2, f2) + F1:$

> $n1 := MatrixVectorMultiply(R2, n2) + CrossProduct(p1, f1) + N1 + CrossProduct(s1, F1):$

> # Torque at joint 1

> $\tau1 := MatrixVectorMultiply(Transpose(n1), MatrixVectorMultiply(Transpose(R1), z0)):$

> $\tau := Vector([[expand(\tau1)], [expand(\tau2)], [expand(\tau3)], [expand(\tau4)], [expand(\tau5)]]):$

> $with(CodeGeneration):$

> # In view is represented derive of 5 dynamic equations of motion for represented model of 5 Dof robot

> # Joint Torques/Forces

> $Matlab(expand(\tau1), resultname = "Eq\_1"):$

Eq_1=...

> $Matlab(expand(\tau2), resultname = "Eq\_2"):$

Eq_2=...

> $Matlab(expand(\tau3), resultname = "Eq\_3"):$

Eq_3=...

> $Matlab(expand(\tau4), resultname = "Eq\_4"):$

Eq_4=...

> $Matlab(expand(\tau 5), resultname = \text{"Eq\_5"})$:

```
Eq_5 = m5 * s5y * cos(q4) * ddq3 - m5 * cos(q4) ^ 2 * dq3 ^ 2 * s5z - m5 * s5x * sin(q3) * ddq1 + m5 * L34 *
cos(q4) * ddq3 + m5 * cos(q4) * cos(q3) * ddd2 + m5 * cos(q4) * cos(q3) * g - m5 * d5 * cos(q4) ^ 2 * dq3 ^ 2
+ 0.2e1 * m5 * dq1 * s5z * dq3 * cos(q3) * sin(q4) * cos(q4) + 0.2e1 * m5 * d5 * dq1 * dq3 * cos(q3) * sin(q4)
* cos(q4) + m5 * dq1 ^ 2 * s5y * cos(q4) * cos(q3) * sin(q3) + m5 * L34 * dq1 ^ 2 * cos(q4) * cos(q3) * sin(q3)
- m5 * dq1 ^ 2 * s5x * sin(q4) * cos(q4) * cos(q3) ^ 2 + 0.2e1 * m5 * dq1 * s5x * dq3 * cos(q3) * cos(q4) ^ 2 +
0.2e1 * m5 * sin(q3) * dq1 * s5y * sin(q4) * dq3 + 0.2e1 * m5 * L34 * sin(q3) * dq1 * sin(q4) * dq3 + m5 * d5
* dq1 ^ 2 * cos(q4) ^ 2 * cos(q3) ^ 2 + m5 * cos(q4) * dq3 ^ 2 * s5x * sin(q4) - 0.2e1 * m5 * s5x * cos(q3) *
dq1 * dq3 - 0.2e1 * m5 * sin(q3) * dq1 * s5z * dq4 - 0.2e1 * m5 * d5 * sin(q3) * dq1 * dq4 - m5 * s5y * ddq1 *
sin(q4) * cos(q3) - m5 * L34 * ddq1 * sin(q4) * cos(q3) + m5 * dq1 ^ 2 * s5z * cos(q4) ^ 2 * cos(q3) ^ 2 + m5 *
ddd5 - m5 * d5 * dq4 ^ 2 - m5 * s5x * ddq4 - m5 * dq1 ^ 2 * s5z - m5 * s5z * dq4 ^ 2 - m5 * d5 * dq1 ^ 2;
```

> Ende of Maple file.

5 CONCLUSIONS

Based on presented paper can be concluded that Newton-Euler formulations are very useful for dynamical modelling of systems generally and robotic systems especially.

Use of Maple software is very useful for modelling of complex systems and representations of results symbolically – representation of expressions of dynamical model with many characters (up to 100,000 characters).

Opportunity of Maple software to convert derived expressions for Matlab use is very helpful for future analyses and simulations of proposed robot system.

REFERENCES

[1] J. Y. S. Luh, M. W. Walker, R. P. Paul. On-Line Computational Scheme for Mechanical Manipulators. *Journal of Dynamic Systems, Measurement, and Control* **1980** (102), No. 2, 69 - 76.

[2] M. Brady. et al. Robot Motion: Planning and Control. MIT Press, Cambridge, MA., **1982**.

[3] P. Khosla. Estimation of Robot Dynamics Parameters: Theory and Application. Institute for Software Research. Paper 651, Carnegie Mellon University. **1987**.

[4] A. Shala, R. Likaj. Design of Genetic Algorithm for optimization of Fuzzy Neural Network Controller. *8th International Conference Modern Technologies in Manufacturing, Cluj-Napoca, Romania.* **2007**.

[5] H. K. Dave, K. P. Desai, H. K. Raval. Investigations on prediction of MRR and surface roughness on electro-discharge machine using regression analysis and artificial neural network programming. *Proceedings of the World Congress on Engineering and Computer Science* **2008**, 123 - 128.

[6] K. Frydrýšek, R. Jančo. Simple Planar Truss (Linear, Nonlinear and Stochastic Approach). *Journal of Mechanical Engineering – Strojnícky časopis* **2016** (66), No. 2, 5-12.

[7] R. Gogola, J. Murín, J. Hrabovský. Numerical Calculation of Overhead Power Lines Dynamics. *Journal of Mechanical Engineering – Strojnícky časopis*, **2016** (66), No. 2, 13 - 22.

[8] R. Jančo, L. Écsi, P. Élesztős. Fsw numerical simulation of aluminium plates by sysweld - PART I. *Journal of Mechanical Engineering – Strojnícky časopis* **2016** (66), No. 1, 47 - 52.

[9] A. Shala, M. Bruqi. Trajectory Tracking of Mobile Robot using Designed Optimal Controller, *International Journal of Mechanical Engineering and Technology* **2017** (8), No. 8, 649 – 658.

[10] A. Shala, X. Bajrami. Dynamic analysis of multi-body mechanism using vector loops. *International Journal of Civil Engineering and Technology* **2017** (8), No. 9, 1084 – 1092.

[11] J. Danko, T. Milesich, J. Bucha. Nonlinear Model of the Passenger Car Seat Suspension System. *Journal of Mechanical Engineering – Strojnícky časopis* **2017** (67), No. 1, 23 - 28.