Bartłomiej FAJDEK, Krzysztof JANISZOWSKI
INSTYTUT AUTOMATYKI I ROBOTYKI PW

Application of the PExSim to Modelling and Simulation of Robot Manipulators

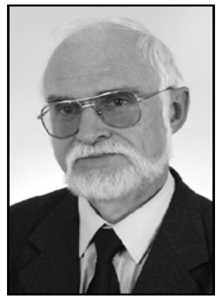
Mgr inż. Bartłomiej FAJDEK

Autor ukończył studia na kierunku Automatyka i Robotyka na Wydziale Mechatroniki Politechniki Warszawskiej. Zainteresowania naukowe autora obejmują szereg zagadnień z dziedziny robotyki, ze szczególnym uwzględnieniem modelowania oraz symulowania manipulatorów.

e-mail: B.Fajdek@mchtr.pw.edu.pl

Prof. dr hab. inż. Krzysztof JANISZOWSKI

Działalność naukowa w zakresie identyfikacji, modelowania i sterowania procesów. Obszary badań aplikacyjnych: identyfikacja procesów przemysłowych, synteza algorytmów sterowania dla pneumatycznych układów pozycjonujących, hydraulicznych napędów wyporowych, zespołu napędu hybrydowego o 2 stopniach swobody, oraz pakiety oprogramowania dla identyfikacji układów dynamicznych (IDCAD), i modelowania, symulacji działania i sterowania procesów przemysłowych (PExSim).



e-mail: kjanisz@mchtr.pw.edu.pl

Abstract

The paper presents the Robotic Elements plugin for the PExSim which provides many functions that are required in robotics, and addresses areas such as kinematics, dynamics, and trajectory generation. The plugin is useful for research, modelling and simulation of robot manipulators. It can also be a powerful tool for education. A short presentation of blocks that are the part of the plugin, is included. Also a simple example of application is presented.

Keywords: robotic, manipulator, modelling, simulation.

Zastosowanie pakietu PExSim do modelowania oraz symulacji ruchu manipulatorów

Streszczenie

W niniejszym opracowaniu przedstawiony jest plugin Robotic Elements dla pakietu PExSim, który dostarcza wiele funkcji niezbędnych w dziedzinie robotyki oraz nauk z nią związanych takich jak kinematyka, dynamika oraz planowanie trajektorii. Przygotowany zestaw bloków stanowi użyteczne narzędzie do badań, modelowania oraz symulowania manipulatorów. Może zostać także wykorzystany do celów edukacyjnych z dziedziny robotyki. W referacie zawarty jest opis poszczególnych bloków, które wchodzą w skład plugin-u. Ponadto przedstawiono przykład praktycznego zastosowania.

Słowa kluczowe: robotyka, manipulator, modelowanie, symulacja.

1. Introduction

PExSim (Process Explorer and Simulator) [1] is a powerful package that can be used for emulation of complex dynamic systems, composed of predefined dynamic blocks or corresponding multivariable models (estimated with their own procedures), can cooperate in real time with industrial environment and will be easy and flexible to extend by the user writing its own plugin objects. The package is written in C++ and can be run on Windows or Unix (calculation runtime only) platform.

The ***Robotic Elements*** plugin provides many of the essential blocks necessary for modelling and simulation of robot manipulators. Two well known robot manipulators dynamic models are implemented: two – link planar manipulator model and three – link manipulator model. The decision of modeling those kinds of manipulators was justified by their widespread use within industry. For example, two-link planar manipulator model is the second and third arm in PUMA 560 manipulator. They are the global part of the manipulator and they carry out transportation tasks (in contrast to the local part which is usually responsible for orientation of the tool). In this way, this structure may cause the greatest errors of the control process. The method based on the *Lagrange formulation* [2] is used for derivation of equations of motion of manipulators in joint space and Cartesian coordinates.

The plugin provides blocks for manipulating and converting between datatypes such as inverse kinematic, forward kinematic, geometric Jacobian, inverse Jacobian and transpose Jacobian. The trajectory generation block is also included.

Using those blocks, we can easily simulate common robotic motion control strategies such as:

- PD control
- PD control with gravity compensation
- Inverse dynamic control

in joint space and Cartesian coordinates.

This paper is divided into four main sections. In *Section 2* blocks that are included in the plugin are presented with some elementary robotic theory. In *Section 3*, there is an example of the ***Robotic Elements*** plugin application to the motion control of manipulators. In Section 4, there is a short summary.

2. Description of Robotic Elements blocks

Blocks that are part of the ***Robotic Elements*** plugin can be divided into three main groups according to their functions:

- Kinematics blocks
- Trajectory planning blocks
- Dynamic blocks

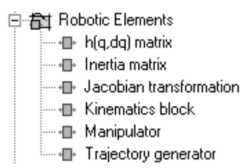


Fig. 1. List of the *Robotic Elements* blocks in the PExSim
Rys. 1. Lista bloków *Robotic Elements* w pakiecie PExSim

The first group of blocks is used for computing the kinematics and differential kinematics of robot manipulators.

In order to manipulate an object in space, it is necessary to describe the position and orientation of the end-effector. The direct kinematics [3] is the problem of solving the Cartesian position and orientation of the end-effector given the knowledge of the kinematic structure of the robot manipulator and the joint coordinates. For n-link robot manipulator, direct kinematics function can be expressed as homogenous transformation:

$$\mathbf{T}_0^n(\mathbf{q}) = \mathbf{A}_1^0(q_1)\mathbf{A}_2^1(q_2)\ldots\mathbf{A}_n^{n-1}(q_n) \\ \mathbf{K}(\mathbf{q}) \qquad (1)$$

where:

$$\mathbf{A}_i^{i-1}(q_i) \quad \text{(for i = 1,…,n)},$$

homogenous transformation matrix relating the description of a point in Frame *i* to the description of a point in Frame *i-1* (describes the relationships between the Cartesian coordinate frames in terms of the Cartesian translation and orientation).

The *Denavit – Hartenberg* convention is used for the construction of the direct kinematics function by composition of the individual coordinate transformations $\mathbf{A}_i^{i-1}(q_i)$ into one homogenous transformation matrix as in (1).

A *Kinematic block* (Fig. 2) is implemented to compute the direct kinematics of two-link – planar manipulator and three – link manipulator. In the *Kinematic block,* we have to define two main parameters: type of manipulator and link lengths.

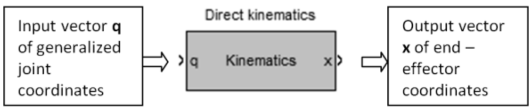


Fig. 2. Direct kinematics block
Rys. 2. Blok kinematyki prostej

The inverse kinematics [3] is the problem of solving joint variables corresponding to a given end – effector position and orientation. It is much more complex than the direct kinematics. First of all, equations to solve are in general nonlinear. Also multiple solutions or no solution (i.e. if the specified transform describes a point out of reach of the manipulator) may exist. For any n-link robot manipulator, inverse kinematics can be derived symbolically [4]. However, in some cases it can be difficult to obtain a closed – form solution. Then numerical solution [5] techniques should be applied. In the ***Robotic Elements*** plugin, *algebraic solution* technique is used to derive the inverse kinematics of two-link – planar manipulator and three – link manipulator. The main reason of choosing this technique was the fact that solutions for those manipulators are well described in literature [4, 5, 6]. In the plugin, the inverse kinematics function is used for defining the initial conditions in the motion equations.

The other crucial issue in modelling and simulation of the robot manipulators is differential kinematics [3]. It gives the relationship between velocities in joint space and corresponding end – effector linear and angular velocity. It is realized by the manipulator geometric Jacobian matrix $\mathbf{J}_G$. It is possible to compute the Jacobian matrix via differentiation of the direct kinematics function. Then it is called the analytical Jacobian ($\mathbf{J}_A$) and we use it when it is necessary to refer to differential quantities of variables defined in the Cartesian coordinates. For n-link manipulator, the end – effector velocity in Cartesian space can be expressed as:

$$\dot{\mathbf{x}} = \mathbf{J}_A(\mathbf{q})\dot{\mathbf{q}} \tag{2}$$

where: $\dot{\mathbf{x}}$ - vector of end - effector velocities in the Cartesian coordinates (*n x 1*), $\dot{\mathbf{q}}$ - vector of joint velocities (*n x 1*), $\mathbf{q}$ - vector of generalized joint coordinates describing the pose of the manipulator (*n x 1*).

The analytical Jacobian is also very useful for designing control schemes in the Cartesian space. Many control schemes require the inverse, transpose and derivative of the Jacobian. For example, we use those transformations in the inverse kinematics algorithm [3].

In the ***Robotic Elements*** plugin, those transformations are implemented in a block called the Jacobian transformation (Fig. 3, 4, 5, 6). From the block options we can choose the type of the transformation and the type of the manipulator (two – link planar manipulator or three – link manipulator).

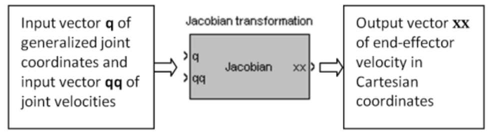


Fig. 3. Analytical Jacobian block
Rys. 3. Blok Jakobianu analitycznego

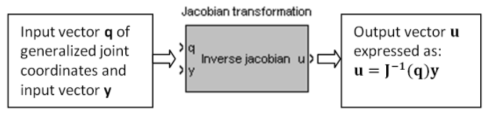


Fig. 4. Inverse Jacobian block
Rys. 4. Blok Jakobianu odwrotnego

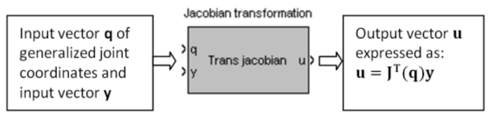


Fig. 5. Transpose Jacobian block
Rys. 5. Blok Jakobianu transponowanego

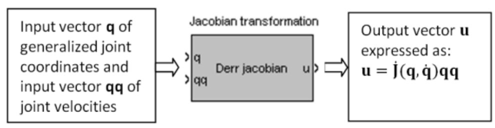


Fig. 6. Transpose Jacobian block
Rys. 6. Blok Jakobianu transponowanego

A second group of blocks is used for computing the dynamic models of the robot manipulators.

An exact dynamic model of the robot manipulator is required for the model-based control and simulation of the robot manipulator motion. To describe the manipulator motion, it is necessary to know the kinematic structure (link lengths, type of joints - rotational or translational) and link inertial parameters of every link (mass, position of center of mass, moment of inertia). There are two main methods for computing the equations of motion. The first method is based on the *Lagrange formulation*. In this method equations of motion are derived starting from the total Lagrangian of the system. It requires the computing the total kinetic and potential energy of the mechanical system. The second method is based on the *Newton – Euler formulation* [7]. This formulation is based on the balance of all forces acting on the generic link of the robot manipulator. It is an inherently recursive method that is computationally efficient. For n – link robot manipulator, the equations of motions can be expressed as:

$$\mathbf{D}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q}) + \mathbf{T_f}(\dot{\mathbf{q}}) = \mathbf{T} \tag{3}$$

where: $\mathbf{q}$ - vector of generalized joint coordinates describing the pose of the manipulator (*n x 1*), $\dot{\mathbf{q}}$ - vector of joint velocities (*n x 1*), $\ddot{\mathbf{q}}$ - vector of joint accelerations (*n x 1*), $\mathbf{T}$ - vector of joint torques which are needed to generate the motion (*n x 1*), $\mathbf{D}(\mathbf{q})$ - inertia matrix (inertia tensor) of the manipulator in joint space (*n x n*), $\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})$ - matrix that describes the Coriolis and centrifugal effects (*n x n*), $\mathbf{G}(\mathbf{q})$ - vector that describes the gravity forces (*n x 1*), $\mathbf{T_f}(\dot{\mathbf{q}})$ - vector that describes the viscous and Coulomb friction (*n x 1*).

In the ***Robotic Elements*** plugin, equations of motion are derived using the *Lagrange formulation* for two – link planar manipulator and three – link manipulator.

In the *Manipulator block* (Fig. 7)*,* we have to define the type of manipulator, kinematics and inertial parameters of the manipulator. The special user interface is made to enter the required parameters (Fig. 8).

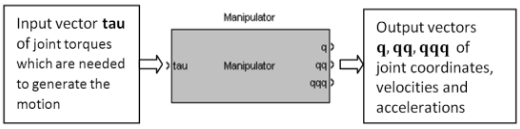


Fig. 7. Manipulator block
Rys. 7. Blok Manipulator

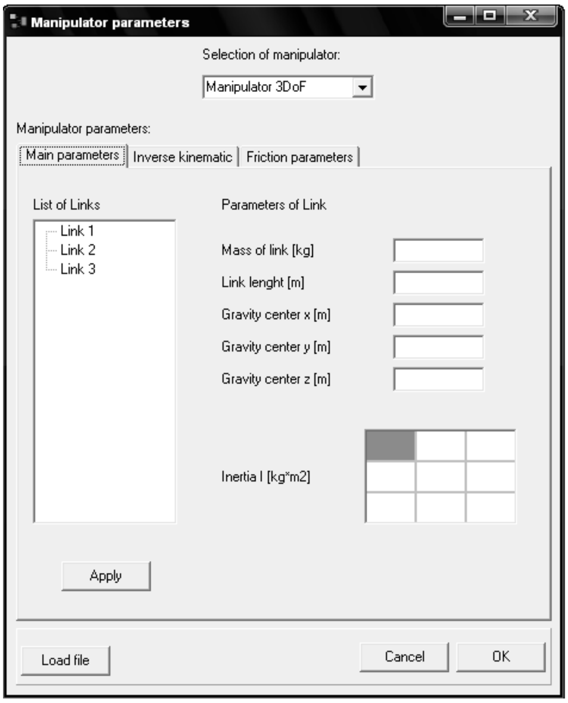


Fig. 8. User interface of the Manipulator block
Rys. 8. Interfejs użytkownika dla bloku Manipulatora

For specified type of manipulator for every link, we can define: length, mass, coordinates of mass center, moment of inertia. Also we can determine the friction model:

- Coulomb friction,
- Coulomb friction plus viscous friction.

There is also a possibility to define initial conditions for the equations of motion. It is very useful if one wants to avoid the kinematics singularities of the manipulator at the start of the simulation.

To allow us the simulations of control diagrams which are based on the model of the robot manipulator, two useful dynamic blocks are implemented in the plugin: *Inertia matrix block* (Fig. 9) and *h(q,qq) matrix block* (Fig. 10).

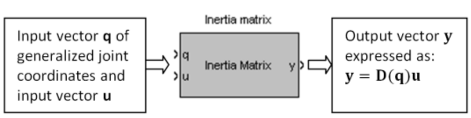


Fig. 9. Inertia matrix block
Rys. 9. Blok macierzy Inercji

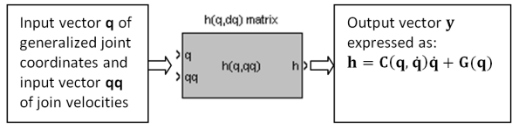


Fig. 10. h(q,qq) matrix block
Rys. 10. Blok macierzy h(q,qq)

Those blocks can be used, for example, to implement *computed torque* control algorithm. A short description of this control method is presented in Section 3.

The other very important thing in the robot manipulator simulation is the trajectory planning. Generally the main task for the robot manipulator is to move from one point to another with the best possible accuracy. The goal of the trajectory planning is to compute desire inputs to the motion control system which ensure that the robot manipulator will follow a specified trajectory. The trajectory can be defined in joint spaces or in Cartesian coordinates. Planning trajectory in Cartesian space is much more complex than in the joint space, especially when there are constraints on the end – effector velocity, joint acceleration, and the trajectory error. Trajectories planned without proper consideration of these constraints often result in poor performance (overshoots, undue velocity fluctuations) [8].

In our approach, the trajectory planning is done by generating the analytical motion primitive and the relative trajectory in a punctual way [3]. A special block called the *Trajectory generation* is implemented to compute the trajectory in the Cartesian space.

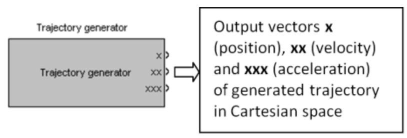


Fig. 11. Trajectory generation block
Rys. 11. Blok do generowania trajektorii

Using this block, we can generate trajectory in two or three dimensions with the trapezoidal velocity profile. We have to define the initial and final point, time of motion and maximal velocity. The special user interface is made to enter the required parameters and verify correctness of computed trajectory (Fig. 12).

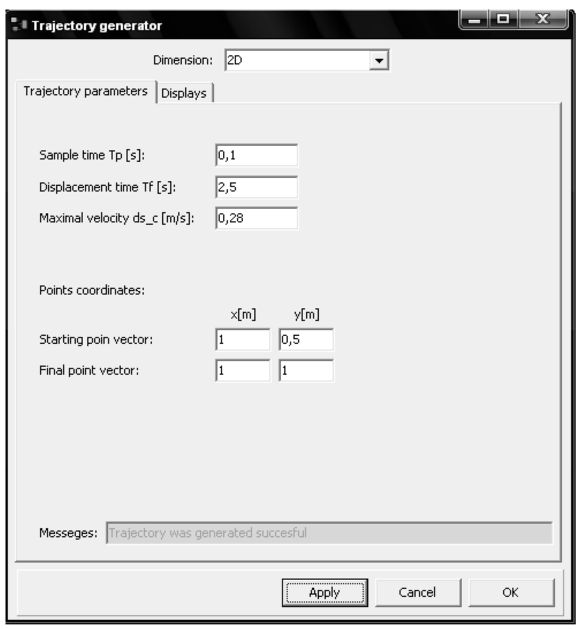


Fig. 12. User interface of the Trajectory generation block
Rys. 12. Interfejs użytkownika dla bloku Generator Trajektorii

3. Application of the Robotic Elements plugin

In this section, a simple example of usage is presented to show main advantages of using the ***Robotic Elements*** plugin in modelling and simulation of the robot manipulators. As an example, the *Computed torque* motion control of three – link robot manipulator is presented.

The Computed torque is well known motion control strategy [9, 10]. It is based on the inverse of the dynamic model of the robot manipulator. The approach is founded on the idea to find a feedback signal that cancels the effects of gravity, friction, the manipulator inertia tensor, Coriolis and centrifugal forces. This signal can be expressed as:

$$\mathbf{u} = \mathbf{D}(\mathbf{q})\mathbf{y} + \mathbf{h}(\mathbf{q}, \dot{\mathbf{q}}) \tag{4}$$

where:

$$\mathbf{h}(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q}) + \mathbf{T_f}(\dot{\mathbf{q}}) \tag{5}$$

The vector **y** represents a control signal and can be expressed as:

$$\mathbf{y} = \mathbf{J}_A^{-1}(\mathbf{q})(\ddot{\mathbf{x}}_\mathbf{d} + \mathbf{K_v}\dot{\mathbf{E}} + \mathbf{K_p}\mathbf{E} - \dot{\mathbf{J}}_A(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}}) \tag{6}$$

where: $\ddot{\mathbf{x}}_\mathbf{d}$ - vector of desired end - effector accelerations in Cartesian coordinates (*n x 1*), $\dot{\mathbf{E}}$ - vector of errors between desired and measured velocities in Cartesian coordinates (*n x 1*), $\mathbf{E}$ - vector of errors between desired and measured positions in Cartesian coordinates (*n x 1*), $\mathbf{K_p}, \mathbf{K_v}$ - diagonal gain matrices for position and velocity loops (*n x n*).

The block diagram implemented in the ***PExSim*** is illustrated in Fig. 13.

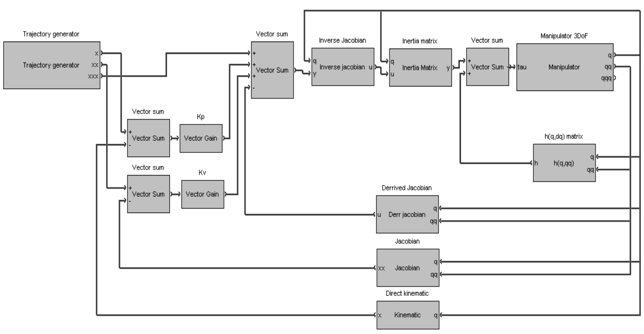


Fig. 13. Block diagram implemented in the inverse dynamic control in the Cartesian space
Rys. 13. Schemat blokowy układu sterowania przy pomocy modelu odwrotnego w przestrzeni Kartezjańskiej

The parameters of the simulated robot manipulator are presented in Tab. 1.

Tab. 1. Manipulator parameters
Tab. 1. Parametry manipulatora

Manipulator parameters	Value[unit]
Lengths of links	1[*m*]
Masses of links	5[*kg*]
Moments of inertia (relative to the centres of mass)	1[kgm^2]

The desired end – effector trajectory has a typical trapezoidal profile and path is a motion from the initial point $x_i = [0;0.8;1]$ to the final point $x_f = [0.5;0.4;1.2]$. The simulation time is of 2.5[*s*] and the maximum velocity of 0.3[*m/s*]. The diagonal gain matrices are defined as follows: $\mathbf{K_v} = diag[16;16;16]$ and $\mathbf{K_p} = diag[100;100;100]$.

In order to demonstrate advantages and disadvantages of the *Computed torque* control strategy, three experiments were carried out:

I Experiment	inverse dynamic model is implemented with exact knowledge of robot manipulator parameters
II Experiment **III Experiment**	inverse dynamic model is implemented with uncertainties in the robot manipulator parameters. The arm is supposed to carry a load: $\Delta m_L = 2[kg]$ (*II experiment*) $\Delta m_L = 5[kg]$ (*III experiment*)

As a result of those experiments we obtained trajectories of motion in Cartesian coordinates.

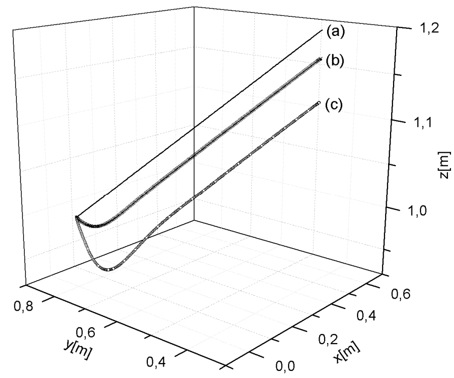


Fig. 14. Trajectories of motion during the simulation (a) I Experiment, (b) II Experiment, (c) III Experiment
Rys. 14. Przebiegi trajektorii ruchu dla przeprowadzonych symulacji: (a) I Eksperyment, (b) II Eksperyment, (c) III Eksperyment

As we can see (Fig. 14), the implementation of the inverse dynamic control strategy indeed requires that that parameters of the robot manipulator dynamics model are accurately known. Then it is possible for the controller to follow the desired trajectory. If we have an imperfect knowledge of the robot manipulator parameters, it is impossible to fully compensate the dynamics of the system. As a result we have poor performance of the controller. Uncompensated mass added to the end – effector causes the trajectory tracking errors.

In practice, robot manipulators face many uncertainties in their dynamic models, in particular the parameters describing the unknown grasped payloads, as well as the unknown frictional coefficients. To compensate for these uncertainties, many researchers have proposed various adaptive control and robust control strategies [11, 12, 13].

4. Summary

This paper has demonstrated the main features of the ***Robotic Elements*** plugin for the ***PExSim*** package. From the short description of blocks that are the part of the plugin, we can see that it provides many essential tools necessary for modelling and simulation of robot manipulators. The plugin is flexible enough for the user to try their own algorithms.

Also such a modular toolbox should close the gap between research in robotics and current teaching. It can be applied in teaching, demonstrating, exercises and lab classes at universities. This will improve and increase the attractiveness of teaching process.

In the near future, new research will be done to extend the plugin capabilities. We plan to implement some new common robotic control strategies, such as: force control, hybrid position – force control and sliding mode control.

5. References

[1] K.Janiszowski, P,Wnuk PExSim – A Package for Investigation of Complex Dynamic Systems, Automation 2007, 2007.

[2] A.K. Bejczy, Robot Arm Dynamics and Control, Jet Propulsion Laboratory, California Institute of Technology, 1974.

[3] B.Siciliano, L.Sciaviacco, Robotics Modelling, Planning and Control, Springer-Verlag, London, December 2008.

[4] R. Featherstone, Position and Velocity Transformations Between Robot End - effector Coordinates and Joint Angles, International Journal of Robotics Research, 1983.

[5] A.A. Goldenberg, B. Benhabib, R.G. Fenton, A Complete Generalized Solution to the Inverse Kinematics of Robots, IEEE Journal of Robotics and Automation, 1985.

[6] J.J. Craig, Introduction to Robotics: Mechanics and Control, Addison – Wesley Publishing Company, Reading, Massachusetts, 1995.

[7] D.E. Orin, R.B. McGhee, M. Vukobratovi´ c, G. Hartoch, Kinematic and Kinetic Analysis of Open-chain Linkages Utilizing Newton–Euler Methods, Mathematical Biosciences, 1979.

[8] S.R. Munasinghe, M. Nakamura, Industrial Robotics: Theory, Modelling and Control, Pro Literatur Verlag, December 2006.

[9] B.R. Markiewicz, Analysis of the Computed Torque Drive Method and Comparison with Conventional Position Servo for a Computer-Controlled Manipulator, Pasadena, 1973.

[10] P.K. Khosla, T. Kanade, Experimental Evaluation of Nonlinear Feedback and Feedforward Control Schemes for Manipulators, International Journal of Robotics Research, 1988.

[11] A. Balestrino, G. De Maria, L. Sciavicco, An Adaptive Model Following Control for Robotic Manipulators, ASME Journal of Dynamic Systems, Measurement and Control, 1983.

[12] M.W. Spong, R. Ortega, R. Kelly, Comments on Adaptive Manipulator Control: A Case Study, IEEE Transactions on Automatic Control, 1990.

[13] S. Nicosia, P. Tomei, Model Reference Adaptive Control Algorithms for Industrial Robots, Automatica, 1984.

[14] P.I.Corke, A Robotics Toolbox for MATLAB, IEEE Robotics and Automation, 1996.

Artykuł recenzowany