Dynamics of a General Multi-axis Robot with Analytical Optimal Torque Analysis

Atef A. Ata, Mohamed A. Ghazy, and Mohamed A. Gadou
Department of Engineering Mathematics and Physics, Faculty of Engineering, Alexandria University, Alexandria 21544, Egypt
Email: atefa@alexu.edu.eg, {mohghazy, eng.mohamed.aref}@gmail.com

Abstract—Robot dynamics is considered one of the most important issues in robot design and control. Many techniques were developed to find equations of motion. One of these techniques is Lagrange-Euler method which is suitable for numerical simulation. In this paper an implementation of Lagrange-Euler to find equations of motion for any general multi-axis robot giving only robot configurations is introduced. The program is verified for a 3 Degree-of-Freedom robot. The robot equations of motion are obtained from the implemented program and verified against those obtained using only Lagrange equation. The output of program for the 3 DOF robot was used to find the optimal torque using analytical optimization analysis for a given set of parameters. This procedure analysis can be used as a benchmark analysis for any optimization technique.

Index Terms—dynamics, lagrange-euler, genetic algorithm, trajectory planning, optimal.

I. INTRODUCTION

Optimal trajectory planning is regarded as a very important area for research where some constraints and objectives are required to be optimized. Some examples of objective functions are minimum path for a manipulator travel to achieve its target, minimum time in travel, and minimum applied torques on manipulator joints. Also there may be constraints on the maximum torque that can be applied on any joint. Robot dynamic model is important as it provides relationship between the applied forces/torques and the motion of robot manipulator.

Many techniques have been developed to find the equations of motion of a multi-degree-of-freedom robot such as Newton-Euler and Lagrange-Euler. A short review in the field of this research including fundamental work and present techniques can be found in Featherstone and Orin [1]. Hollerbach [2] proved that Newton-Euler approach can be formulated as a recursive structure which can be faster than treating the manipulator as a whole; he also showed that Lagrange-Euler can be used in a recursive manner. Lagrange-Euler technique is so suitable for numerical solution. Khalil [3] provided more details about these techniques and presented some techniques on conversion between Cartesian and generalized coordinates. In the comparison between Newton-Euler and Lagrange-Euler Silver [4] showed that the computational complexities of the two techniques are the same.

In this paper, an implementation based of Lagrange-Euler technique to determine the equations of motion for an n-axis robot is presented. An example for a 3 DOF robot is illustrated to verify the proposed algorithm. An analytical optimization approach is investigated as a benchmark for minimum energy using any optimization technique. The paper is organized as follows: section II contains the equations of motion in compact form and details of the algorithm to get each term. Section III presents the case study for a 3 DOF robot while section IV is devoted to analytical optimization analysis followed by discussion and conclusions in section V and references.

II. DYNAMICAL ANALYSIS

The objective of Lagrange-Euler method is to get equations of motion for a robot provided that robot configurations and the kinematic equation in terms of Denavit-Hartenberg are given. Lagrange-Euler technique depends on finding kinetic energy of a body which is changed with its spatial and angular velocity in general motion and finding its potential energy. In the case of rigid body dynamics, the only source for potential energy is gravity. It is suitable for rigid robot but there were some research activities on using this technique for flexible robotic manipulators and this is mentioned by Lin and Yuan [5].

The general form of robot equation of motion can be given using Lagrange-Euler technique in the following form:

$$T_i = \sum_{j=1}^{n} D_{ij} \dot{\theta}_j + I_{(act)} \ddot{\theta}_i + \sum_{j=1}^{n} \sum_{k=1}^{n} D_{ijk} \dot{\theta}_j \dot{\theta}_k + D_i$$  \hspace{1cm} (1)

where $T_i$ is the actuator torque of joint i, $I_{(act)}$ is the actuator inertia of joint i, $q_i$ is the generalized coordinate represented by p if the joint is prismatic or represented by $\theta$ if the joint is revolute, n is the total number of links and D terms can be gotten as follows:

$$D_{ij} = \sum_{p=max(i,j)}^{n} \text{Trace} \left( U_{ip} J_p U_{pj}^T \right)$$  \hspace{1cm} (2)
\[ D_{ij} = \frac{\sum_{p=1}^{n} m_p \mathbf{g}^T \mathbf{U}_p \mathbf{r}_p}{\sum_{p=1}^{n} m_p} \]  

\[ D_{ij} = \sum_{p=1}^{n} -m_p \mathbf{g}^T \mathbf{U}_p \mathbf{r}_p \]  

where \( J_i \) is the pseudo inertia matrix for link \( i \) and is defined by:

\[
J_i = \begin{bmatrix}
-I_{x_i} + I_{y_i} + I_{z_i} & I_{y_i} & I_{z_i} & m_{x_i} \\
I_{x_i} & -I_{x_i} + I_{y_i} + I_{z_i} & I_{z_i} & m_{y_i} \\
I_{x_i} & I_{y_i} & -I_{x_i} + I_{y_i} + I_{z_i} & m_{z_i} \\
m_{x_i} & m_{y_i} & m_{z_i} & m_i
\end{bmatrix}
\]

where \( m_i \) is the mass of link \( i \) and \( x_i, y_i, z_i \) are coordinates of center of link \( i \) relative to the link coordinate frame. On the other hand \( U \) terms can be defined:

\[ U_p = \mathbf{A}_p^T \mathbf{A}_p \mathbf{g} \]

where \( \mathbf{A}_p \) is the transformation matrix of link \( j \), \( m_p \) is the mass of link \( p \), \( \mathbf{g}^T \) is the transpose of gravity matrix and \( \mathbf{r}_p \) is the position vector of center of gravity of link \( p \) relative to its coordinate frame.

In the above equations of motions there are three types of terms. The first type involves the second derivative of the generalized coordinates, it is called Angular Acceleration Inertia term. The second type of terms is quadratic terms in the first derivative of \( q \); these terms are divided into two subtypes. The terms involving product of the type \( \dot{q}_i \dot{q}_j \) are called Centrifugal terms, while those involving product of type \( \dot{q}_i \dot{q}_j \) where \( i \neq j \) are called Coriolis terms. For more details the reader is referred to [6] and [7].

The developed program is able to find equations of motion for any general multi-axis robot giving robot configuration and its \( \mathbf{A} \) matrices. Table I describes program inputs in details.

Program main units are:

A. Model

The model is the main procedure of the program. It is used to do summations and to calculate D terms which are needed to be computed.

B. GetD2

This module is used to calculate terms \( D_{ij} \).

C. GetD3

This module is used to calculate terms \( D_{ijk} \).

D. GetD1

This module is used to calculate terms \( D_i \).

The following is a flowchart for the main component of the implemented program.

**TABLE I. MAIN FEATURES OF THE PROPOSED CODE MODULE.**

<table>
<thead>
<tr>
<th>Input</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( T )</td>
<td>This variable is a matrix which results from concatenation of all ( \mathbf{A} ) matrices in one matrix (i.e. ( T = [\mathbf{A}_1, ..., \mathbf{A}_n] ))</td>
</tr>
<tr>
<td>( n )</td>
<td>This variable is number of robot degrees of freedom</td>
</tr>
<tr>
<td>( G )</td>
<td>This variable is concatenation of gravity matrices</td>
</tr>
<tr>
<td>configuration</td>
<td>This is a vector expressing each link type; each element in vector can values 0 and 1, where 0 indicates a revolute joint and 1 a prismatic joint</td>
</tr>
<tr>
<td>symbols</td>
<td>This is a matrix containing robot links masses and lengths</td>
</tr>
</tbody>
</table>

**III. ALGORITHM IMPLEMENTATION**

The program was tested using a three link manipulator with revolute joint and it was verified against analytical solution using Lagrange equation only.

Consider a 3 DOF planar robot arm as shown in Fig. 2. The robot moves in the vertical plane so the gravity effect will be included in the analysis.
The following table introduces the parameters based on Denavit-Hartenberg notations

<table>
<thead>
<tr>
<th>Link #</th>
<th>θ</th>
<th>d</th>
<th>a</th>
<th>α</th>
</tr>
</thead>
<tbody>
<tr>
<td>Link 1</td>
<td>θ₁</td>
<td>0</td>
<td>l₁</td>
<td>0</td>
</tr>
<tr>
<td>Link 2</td>
<td>θ₂</td>
<td>0</td>
<td>l₂</td>
<td>0</td>
</tr>
<tr>
<td>Link 3</td>
<td>θ₃</td>
<td>0</td>
<td>l₃</td>
<td>0</td>
</tr>
</tbody>
</table>

And the following table identifies the model parameters as:

<table>
<thead>
<tr>
<th>Joint variables</th>
<th>θ₁, θ₂, θ₃</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mass of links</td>
<td>m₁, m₂, m₃</td>
</tr>
<tr>
<td>Link parameters</td>
<td>l₁, l₂, l₃</td>
</tr>
</tbody>
</table>

The transformation matrices can be obtained from the parameters as:

\[ A₁ = \begin{bmatrix} c₁ & s₁ & 0 & l₁c₁ \\ s₁ & c₁ & 0 & l₁s₁ \\ 0 & 0 & 1 & 0 \end{bmatrix}, \quad A₃ = \begin{bmatrix} c₃ & s₃ & 0 & l₃c₃ \\ s₃ & c₃ & 0 & l₃s₃ \\ 0 & 0 & 1 & 0 \end{bmatrix} \]

The proposed program will use these matrices along with the robot parameters as inputs and the simulation will be carried out as shown in the flowchart. The required equations of motion for the three links are given in the form:

\[ T₁ = \frac{1}{3} m₁ \left( 3l₁^2 + 6l₁l₂c₂ + 3l₂^2 + 3l₁l₃c₃ + l₃^2 + \frac{1}{3} l₁^3 \right) \ddot{q}_₁ + \frac{1}{3} m₂ \left( 3l₂^2 + 6l₂l₃c₃ + 3l₃^2 + 3l₂l₁c₁ + l₁^2 + \frac{1}{3} l₂^3 \right) \ddot{q}_₂ + \frac{1}{6} m₃ \left( 2l₃ + 3l₃c₃ \right) \ddot{q}_₃ + I_{θ₁} \ddot{θ}_₁ \\
+ \frac{1}{2} m₃ l₁ \left( l₁s₂ - l₃s₃c₃ \right) \dot{q}_₁ \dot{q}_₃ + \frac{1}{2} m₃ l₂ \left( l₂s₃c₁ - l₃s₂c₃ \right) \dot{q}_₂ \dot{q}_₃ + \frac{1}{2} m₃ l₃ \left( l₃s₁c₂ - l₂s₃c₃ \right) \dot{q}_₃ \dot{q}_₃ \\
+ \frac{1}{2} m₃ \left( l₃s₁c₂ + l₂s₃c₃ \right) \ddot{q}_₃ + \frac{1}{2} m₃ l₁ \left( l₁s₂ + l₃s₃c₃ \right) \ddot{q}_₁ + \frac{1}{2} m₃ l₂ \left( l₂s₃c₁ - l₃s₂c₃ \right) \ddot{q}_₂ \]

\[ T₂ = \frac{1}{6} m₁ \left( 6l₁^2 + 6l₁l₂c₂ + 6l₂^2c₂ + 2l₁^2 + 3l₂^2 + 3l₃^2 \right) \ddot{q}_₁ + \frac{1}{6} m₂ \left( 2l₂^2 + 3l₂c₂ \right) \ddot{q}_₂ + \frac{1}{3} m₃ \left( 3l₃^2 + 3l₃c₃ + l₃^2 \right) \ddot{q}_₃ + \frac{1}{3} m₃ \left( l₃c₃ + 3l₃c₃ \right) \ddot{q}_₃ + I_{θ₂} \ddot{θ}_₂ \\
+ \frac{1}{2} m₃ l₁ \left( l₁s₂ + l₃s₃c₃ \right) \dot{q}_₁ \dot{q}_₃ + \frac{1}{2} m₃ l₂ \left( l₂s₃c₁ - l₃s₂c₃ \right) \dot{q}_₂ \dot{q}_₃ + \frac{1}{2} m₃ l₃ \left( l₃s₁c₂ - l₂s₃c₃ \right) \dot{q}_₃ \dot{q}_₃ \\
+ \frac{1}{2} m₃ \left( l₃s₁c₂ + l₂s₃c₃ \right) \ddot{q}_₃ + \frac{1}{2} m₃ l₁ \left( l₁s₂ + l₃s₃c₃ \right) \ddot{q}_₁ + \frac{1}{2} m₃ l₂ \left( l₂s₃c₁ - l₃s₂c₃ \right) \ddot{q}_₂ \]

\[ T₃ = \frac{1}{6} m₁ \left( 6l₁^2 + 6l₁l₂c₂ + 6l₂^2c₂ + 2l₁^2 + 3l₂^2 + 3l₃^2 \right) \ddot{q}_₁ + \frac{1}{6} m₂ \left( 2l₂^2 + 3l₂c₂ \right) \ddot{q}_₂ + \frac{1}{3} m₃ \left( 3l₃^2 + 3l₃c₃ + l₃^2 \right) \ddot{q}_₃ + \frac{1}{3} m₃ \left( l₃c₃ + 3l₃c₃ \right) \ddot{q}_₃ + I_{θ₃} \ddot{θ}_₃ \\
+ \frac{1}{2} m₃ l₁ \left( l₁s₂ + l₃s₃c₃ \right) \dot{q}_₁ \dot{q}_₃ + \frac{1}{2} m₃ l₂ \left( l₂s₃c₁ - l₃s₂c₃ \right) \dot{q}_₂ \dot{q}_₃ + \frac{1}{2} m₃ l₃ \left( l₃s₁c₂ - l₂s₃c₃ \right) \dot{q}_₃ \dot{q}_₃ \\
+ \frac{1}{2} m₃ \left( l₃s₁c₂ + l₂s₃c₃ \right) \ddot{q}_₃ + \frac{1}{2} m₃ l₁ \left( l₁s₂ + l₃s₃c₃ \right) \ddot{q}_₁ + \frac{1}{2} m₃ l₂ \left( l₂s₃c₁ - l₃s₂c₃ \right) \ddot{q}_₂ \]

IV. ALGORITHM IMPLEMENTATION

The final model equations of motions can be used to find torque on each link at any time knowing the prescribed trajectory for each joint. Fourth-order polynomial trajectory with rest-to-rest motion is assumed in the form:

\[ θ(τ) = c₀ + c₁τ + c₂τ² + c₃τ³ + c₄τ⁴ \]

where the coefficients c₀, c₁, c₂, c₃, and c₄ are constants that to be determined from the initial and final conditions. Initial and final positions as well as rest-to-rest motion reduces the unknown coefficients to only one. For the non optimized case, the fourth coefficient vanishes (third-order polynomial) and the initial and final conditions are sufficient to determine the coefficients c₀, c₁, c₂, and c₃. For the optimized case we need to find the fourth coefficient. It is the objective of this analysis to find how this coefficient can optimize the energy consumption for the robot arm. The cost function under consideration can be assumed as:

\[ T = \sqrt{T₁² + T₂² + T₃²} \]

This function was used by Garg and Kumar [8]. To find the optimized value of this coefficient there are many techniques to be used and the most of them are heuristics techniques like Genetic Algorithm, Neural Networks, and Particle Swarm Optimization. The heuristic technique is preferable because of the hardness or even disability to
use gradient techniques which requires a lot of time to find the solution. But there is a need first to ensure that the optimization would get a better solution than the non optimized case so analytical optimization analysis procedure is used here. This analytical optimization technique is a novel efficient way and it is used as a benchmark to ensure the benefit when using any optimization technique.

A small program is implemented in C programming language that takes equations of motion as input and also initial and final desired positions. In this program a loop that tries a range of possible values for this coefficient and evaluates the objective function and this range is changed manually to see the best possible value. Analytical analysis was done for initial position \((\theta_i) = 0\) radian and final position \((\theta_f) = 1\) radian for each link and travelling time of 5 seconds. Each link has mass of 0.5 kg and 1 meter in length. The optimized value for control variable that is obtained from program is -0.17 and the optimized case is compared to non optimized case and the numerical results are shown in figures 3, 4, and 5 respectively. Fig. 3 shows the comparison between the optimized and the optimized case:

![Figure 3. Optimized objective function versus non optimized](image)

While in Fig. 4 the optimized torque for each link during time interval is shown:

![Figure 4. Torques on each link for optimized case](image)

And Fig. 5 shows changing of position, velocity, and acceleration with time in the case of optimized value:

![Figure 5. Optimized trajectory for each joint](image)

V. CONCLUSIONS

An algorithm to find the equations of motion for multi-link robotic arm is presented in this paper. The implemented program can be used to find equations of motions for any robot configuration and this would save a lot of time that will be spent in solving many equations especially when number of DOF is high. Robot equations of motion can be used in optimal trajectory planning and as a benchmark of optimization analytical analysis is carried out for a certain set of parameters to ensure the benefit of optimization, so any optimization technique can be used to find the optimal trajectory planning for any set of given parameter.

REFERENCES


Dr. Atef A. Ata was born in Alexandria, Egypt in 1962 and received his B. Sc. Degree with Honor in Mechanical Engineering from Alexandria University in Egypt in 1985. After his graduation he joined the same university as a Lecturer where he obtained his M. Sc. Degree in Engineering Mathematics (Hydrodynamics) in 1990. In 1996 he obtained his Ph. D in Engineering Mathematics (Robotics) as a Joint-

147
Venture between University of Miami, Florida, USA and Alexandria University in Egypt. He joined Alexandria University again as Assistant Professor till 2001. Then he joined the Mechatronics Engineering Department, International Islamic University Malaysia as an Assistant Professor, Associate Professor (2004) and as a Head of Department (November 2005-June 2007). Currently he is a Professor of engineering mechanics at the Faculty of Engineering, Alexandria University, Egypt. Dr. Atef is a senior member of IACSIT as well as Egyptian Engineering Syndicate. He was also one of the Editorial Consultant Board for 2006-2008 for the International Journal of Advanced Robotic Systems, Austria. Currently he is one of the Editorial board of Mechanical Engineering Research, Canada and an Associate Editor of Alexandria Engineering Journal, Egypt hosted by Elsevier. His research interest includes Dynamic and Control of Flexible Manipulators, Trajectory Planning, Genetic Algorithms and Modelling and Simulation of Robotic Systems.

**Dr. Mohammed A. Ghazy** received the Bachelor of Science in Textile Engineering (Spinning) from University of Alexandria, Egypt with degree of honor in 1998. In 2000 he enrolled in the same institution to pursue the Master of Science degree in Engineering Mathematics which he received in 2004. He was formerly employed with the Department of Engineering Mathematics and Physics, University of Alexandria, as a Teaching Assistant from 2000 to 2004. Mohammed Ghazy was enrolled in 2006 in the Department of Aerospace Engineering, Old Dominion University to pursue the Doctor of Philosophy degree and was awarded a Doctoral Fellowship in August 2009. In May 2010 he received the Batten College of Engineering Excellence Award in Aerospace Engineering, and he graduated on August 2010. From June 2011 till now he is working as an assistant professor at the Department of Engineering Mathematics and Physics, University of Alexandria, Egypt.

**Mohamed A. Gadou** received his B.Sc. Degree with Honor in Computer Systems and Engineering from Alexandria University in Egypt in 2008. In 2009 he was formerly employed as a Teaching Assistant in the Department of Engineering Mathematics and Physics, University of Alexandria. In 2010 he enrolled in the same University to pursue his M.Sc. in Engineering Mathematics. Currently he is preparing for his master thesis.