Robot dynamics: A recursive algorithm for efficient calculation of Christoffel symbols

Mohammad Safeea, Richard Bearee, and Pedro Neto

***Abstract*— Christoffel symbols are very important in robotics. They are used for tuning various proposed robot controllers, for determining the bounds on Coriolis/Centrifugal matrix, for mathematical formulation of optimal trajectory calculation, among others. In literature, Christoffel symbols are calculated from the Lagrangian formulation using an off-line generated symbolic formula. In this study we present an efficient recursive non-symbolic method where Christoffel symbols are calculated based on the robot's transformation matrices and inertial parameters. The proposed method was analyzed in terms of execution time, computational complexity and numerical error. Results show that the proposed algorithm compares favorably with existing methods.**

***Index Terms* — Dynamics, Christoffel symbols, recursive algorithms**

I. Introduction

Dynamics of robots is an important topic since that it is highly involved in their design, simulation and control. Owing to its importance this subject had been studied extensively in the past three decades. Thus, several algorithms and methods had been developed to calculate it [1] [2]. Nevertheless, this subject remains till this day open for extensive research.

The robotics literature has described two formulations for robot dynamics:

1) Operational space formulation. Where the dynamics equations are referenced to the manipulator's end-effector [3]. This formulation is successfully applied for the combined application of motion and force control [4]. Consequently, various operational space algorithms for efficient calculation of robot dynamics have been developed [5] and [6].
2) Joint space formulation. It describes the dynamics of the robot in joint space. This formulation manifests the effect of the joints' positions, velocities and accelerations on the torques and vice-versa.

One of the earliest methods used to deduce the equations of robot dynamics is based on the Lagrangian formulation. This method is well established in the literature [7], and can be encountered in most robotics textbooks. Lagrangian formulation is a straight forward approach that treats the robot as a whole and utilizes its Lagrangian, a function that describes the energy of the mechanical system:

Mohammad Safeea is with the Department of Mechanical Engineering, University of Coimbra, Portugal and with Arts et Métiers, ParisTech, France, e-mail: ms@uc.pt.

Richard Bearee is with Arts et Métiers, ParisTech, Lille, France, e-mail: Richard.BEAREE@ENSAM.EU.

Pedro Neto is with the Department of Mechanical Engineering, University of Coimbra, Coimbra, Portugal, e-mail: pedro.neto@dem.uc.pt.

$$\mathcal{L} = \mathcal{T} - \mathcal{U} \tag{1}$$

Where $\mathcal{L}$ is the Lagrangian function, $\mathcal{T}$ is the kinetic energy and $\mathcal{U}$ is the potential energy, all described in terms of the generalized coordinates $\boldsymbol{q}$. In such a case, the associated generalized forces $\boldsymbol{\tau}$:

$$\boldsymbol{\tau} = \frac{d}{dt}(\frac{\delta \mathcal{L}}{\delta \dot{\boldsymbol{q}}})^{\mathrm{T}} - (\frac{\delta \mathcal{L}}{\delta \boldsymbol{q}})^{\mathrm{T}} \tag{2}$$

Consequently, the canonical form of the inverse dynamics derived using the Lagrangian is:

$$\tau_k = \sum_j a_{kj}\ddot{q}_j + \sum_{i,j} c^k_{ji}\dot{q}_j\dot{q}_i + g_k \tag{3}$$

Where τ_k is the torque at joint k, a_{kj} is the (k, j) element of the mass matrix, $\ddot{q}_j$ is the angular acceleration of joint j, $\dot{q}_j$ is the angular velocity of joint j, g_k is the torque at joint k due to gravity, and c^k_{ji} represent Christoffel symbols, from equation (3) in [8] c^k_{ji} is given by:

$$c^k_{ji} = c^k_{ij} = \frac{1}{2}\left(\frac{\partial a_{kj}}{\partial q_i} + \frac{\partial a_{ki}}{\partial q_j} - \frac{\partial a_{ij}}{\partial q_k}\right) \tag{4}$$

Even though the Lagrangian formulation can be considered as a straight forward approach, the method requires partial differentiation. Despite the fact that symbolic manipulation methods have been utilized to perform the differentiation [9], the method still lacks the efficiency in terms of execution-time. This can be noticed when the robot presents a relatively high number of DOF as noted in [10] and most remarkably in [11], where the author performed comparison of execution-times required to run dynamics simulations based on models derived by Newton-Euler recursive technique and Euler-Lagrange technique. Where it was reported execution-times difference of order of magnitude which put the case in favor of the Newton-Euler recursion method.

Due to their efficiency, researchers developed various recursive algorithms for calculating the inverse dynamics [12], the forward dynamics [13, 14], the joint space inertia matrix [15, 16]. Nevertheless, we are not aware of any proposed recursive method for calculating Christoffel symbols numerically. As such, in this study we present a method for calculating Christoffel symbols recursively. We also analyze the performance of the proposed algorithm in terms of number of operations, execution time and numerical error. MATLAB code of the proposed algorithm, including implementation examples, is available on-line in the repository [17].

II. Motivation and Contribution

Calculating Christoffel symbols is very important in the robotics field. Hence, they have been used for solving various robotics problems. In [18] they are used for calculating the bounds on the Coriolis/Centrifugal matrix. These bounds play an important role for designing and tuning various proposed robot controllers [19, 20, 21, 22, 23, 24]. In addition, Christoffel symbols have been used in [25] for proposing a dynamic neurocontroller of robotic arms. They can also be used to calculate a special form of Coriolis matrix that preserves the skew symmetry property [26] (an essential property for various control algorithms). Christoffel symbols are also important for planning time optimal trajectories and optimal velocity-profile generation [27]. In such a case, the geometric path is parametrized using a vector function of a scalar parameter $\theta(t)$. So that the inverse dynamics equation (which enters the optimization as differential constraint) is reformulated to decouple the configuration dependent coefficients from the time dependent parameter $\theta(t)$, as shown in [27]:

$$\boldsymbol{\tau} = \tilde{\boldsymbol{m}}(\boldsymbol{q})\ddot{\theta} + \tilde{\boldsymbol{c}}(\boldsymbol{q})\dot{\theta}^2 + \tilde{\boldsymbol{d}}(\boldsymbol{q}) \tag{5}$$

In such a case, Christoffel symbols are utilized for calculating the (configuration dependent) coefficients $\tilde{\boldsymbol{c}}(\boldsymbol{q})$ in each configuration on the discretized geometrical path.

In [28] Christoffel symbols are calculated in symbolic form, based on the Lagrangian formulation of the robot dynamics. This method has become the norm and is presented in standard robotics textbooks [29, 26, 30], including the Handbook of Robotics [31] P100, where the deduction of Christoffel symbols is introduced under the section "Lagrange Formulation". Nevertheless, symbolic methods for performing the calculations have major drawbacks:

- The symbolic manipulation of the equations is time consuming, so it has to be performed off-line.
- For high Degrees Of Freedom (DOF), the symbolic equations become very complex resulting in much slower execution times than numerical methods, a fact reported in literature [10].

Apart form that, a recursive method for calculating Christoffel symbols has several advantages over the symbolic:

- Unlike the symbolic methods, recursive methods can be used on-the-fly, and does not require an off-line preprocessing. This makes recursive methods essential for calculating the Christoffel symbols on-the-fly in robots that change its kinematic chain and dynamic model, for example by adding or subtracting extra bodies (including Reconfigurable and Self Assembling robots , or when attaching bodies to EEF).
- Since that recursive methods are used on-the-fly, The proposed recursive method allows updating the dynamical constants (dynamical model) of the robot on-the-fly, this allows the algorithm to use initial estimates of dynamic constants while learning and tuning them more accurately during operation. This can not be done easily in symbolic methods that require off-line code regeneration.

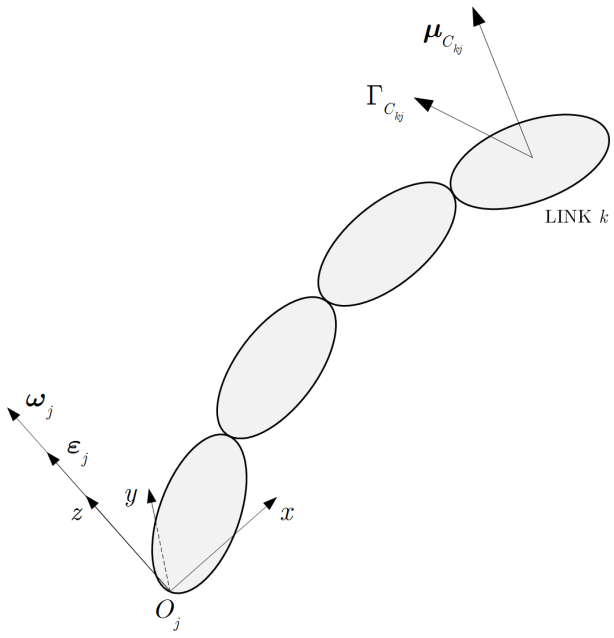

Fig. 1
Inertial moment $\boldsymbol{\mu}_{Ckj}$ and linear acceleration $\ddot{\boldsymbol{p}}_{Ckj}$ of center of mass of link i transferred by frame j

Moreover, the proposed method calculates Christoffel symbols based on the robot's transformation matrices and inertial parameters, without requiring partial differentiation.

Apart from the previously listed computational advantages, the proposed method offers more insight into the nature of Christoffel symbols from the point of view of Newton mechanics.

III. Theory and principals

The proposed algorithm builds on what we call the frame injection principal [16], Figure 1, in which each frame j attached to joint j will transfer to link k a linear acceleration into its center of mass and an inertial moment around its center of mass. In this study we notate them by Γ_{Ckj} and $\boldsymbol{\mu}_{Ckj}$, respectively. This transfer is due to the rotational effect of joint j around its axes of rotation, or the z axis of frame j according to modified Denavit Hartenberg (MDH) designation. This cause and effect relationship between frame j and link k is referred to by the subscript kj in Γ_{Ckj} and $\boldsymbol{\mu}_{Ckj}$, while the C in the subscript is used to refer to the mass center of link k. The same subscript notation will hold throughout this study for denoting frame-link interaction of cause-and-effect unless stated otherwise.

A. Link's acceleration due to the single-frame rotation

Each frame j transfers to link k three acceleration vectors tangential acceleration, normal acceleration and Coriolis acceleration. The first of which is shown in Figure 2, it is due to the angular acceleration of frame j:

$$\Gamma^{\tau}_{Ckj} = \boldsymbol{\varepsilon}_j \times \boldsymbol{p}_{Ckj} \tag{6}$$

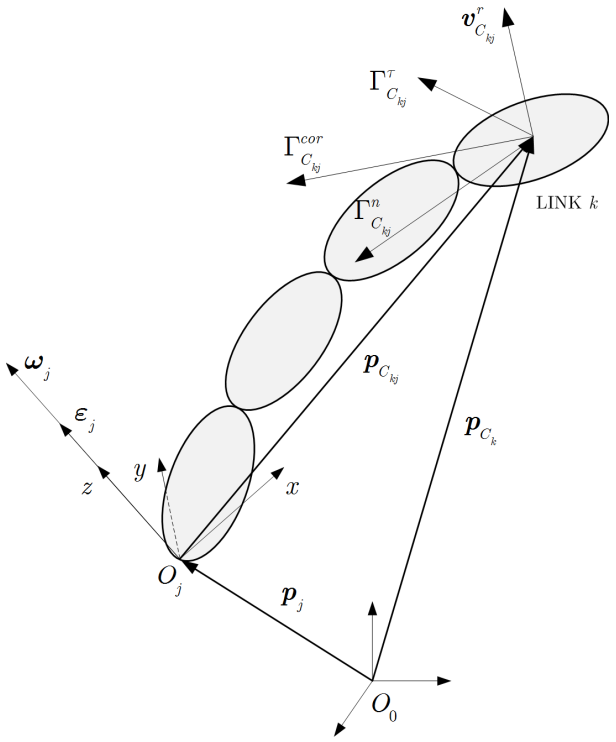

Fig. 2
TANGENTIAL ACCELERATION OF CENTER OF MASS OF LINK k TRANSFERRED BY FRAME j

Where Γ^{τ}_{Ckj} is the tangential acceleration of the center of mass of link k due to the rotation of frame j, the symbol $\times$ is used to denote the cross product (the same notation of the cross product will hold throughout this study) and $\boldsymbol{p}_{Ckj}$ is the vector connecting the origin of frame j and the center of mass of link k. $\boldsymbol{\varepsilon}_j$ is the angular acceleration of link j:

$$\boldsymbol{\varepsilon}_j = \ddot{q}_j \boldsymbol{k}_j \tag{7}$$

Where $\boldsymbol{k}_j$ is the unit vector associated with the z axis of joint j, and $\ddot{q}_j$ is the angular acceleration of that joint.

Concerning the normal acceleration, each frame j transfers to link k a normal acceleration due to its rotation, Figure 2:

$$\Gamma^{n}_{Ckj} = \boldsymbol{\omega}_j \times (\boldsymbol{\omega}_j \times \boldsymbol{p}_{Ckj}) \tag{8}$$

Where $\boldsymbol{\omega}_j$ is the angular velocity of link j due to the rotational effect of joint j. It is given by:

$$\boldsymbol{\omega}_j = \dot{q}_j \boldsymbol{k}_j \tag{9}$$

We can rewrite the equation of the normal acceleration transferred to link k due to frame j by:

$$\Gamma^{n}_{Ckj} = \boldsymbol{k}_j \times (\boldsymbol{k}_j \times \boldsymbol{p}_{Ckj}) \dot{q}_j^2 \tag{10}$$

The third acceleration transferred is Coriolis acceleration, Figure 2, in which each frame j transfers to center of mass of link k Coriolis acceleration Γ^{cor}_{Ckj}:

$$\Gamma^{cor}_{Ckj} = 2\boldsymbol{\omega}_j \times \boldsymbol{v}^r_{Ckj} \tag{11}$$

Where $\boldsymbol{\omega}_j$ is as described previously in equation (9), and $\boldsymbol{v}^r_{Ckj}$ is the velocity transferred to the center of mass of link k from frames $j+1$ up to frame k. Herein $\boldsymbol{v}^r_{Ckj}$, the superscript r is used to denote that this is a relative velocity, and C in the subscript is used to refer to the mass center of link k, so that $\boldsymbol{v}^r_{Ckj}$ can be calculated from:

$$\boldsymbol{v}^r_{Ckj} = \sum_{m=j+1}^{k} \boldsymbol{\omega}_m \times \boldsymbol{p}_{Ckm} \tag{12}$$

The total linear acceleration transferred by frame j to the center of mass of link k is given by:

$$\Gamma_{Ckj} = \Gamma^{\tau}_{Ckj} + \Gamma^{n}_{Ckj} + \Gamma^{cor}_{Ckj} \tag{13}$$

B. Link's inertial moment due to single-frame effect

It can be proved that each frame j will transfer to link k three inertial moments, the first of which is due to angular acceleration of frame j, it is given by:

$$\boldsymbol{\mu}^{\tau}_{Ckj} = (\mathbf{R}_k \mathbf{I}^k_k \mathbf{R}^{\mathrm{T}}_k) \boldsymbol{\varepsilon}_j \tag{14}$$

While $\boldsymbol{\mu}^{\tau}_{Ckj}$ is the moment transferred by frame j into link k due to frame's j angular acceleration, $\mathbf{R}_k$ is the rotation matrix of frame k in relation to base frame, and $\mathbf{I}^k_k$ is 3×3 inertial tensor of link k around its center of mass represented in frame k.

The second inertial moment transferred from frame j to link k is due to centrifugal effect:

$$\boldsymbol{\mu}^{n}_{Ckj} = \frac{1}{2}(\mathbf{L}_k \boldsymbol{\omega}_j) \times \boldsymbol{\omega}_j \tag{15}$$

Where $\mathbf{L}_k$ is a 3×3 matrix that is calculated from:

$$\mathbf{L}_k = \mathbf{R}_k(\mathrm{tr}(\mathbf{I}^k_k)\mathbf{1}_3 - 2\mathbf{I}^k_k)\mathbf{R}^{\mathrm{T}}_k \tag{16}$$

The subscript in $\mathbf{L}_k$ is to notate that the matrix calculated pertains to link k. $\mathrm{tr}(\mathbf{I}^k_k)$ is the trace of the inertial tensor and $\mathbf{1}_3$ is the identity matrix.

The third inertial moment transferred from frame j to link k is due to Coriolis effect:

$$\boldsymbol{\mu}^{cor}_{Ckj} = (\mathbf{L}_k \boldsymbol{\omega}_j) \times \boldsymbol{\omega}^r_{kj} \tag{17}$$

Where $\boldsymbol{\omega}^r_{kj}$ can be calculated from:

$$\boldsymbol{\omega}^r_{kj} = \sum_{m=j+1}^{k} \boldsymbol{\omega}_m \tag{18}$$

Thus, the total inertial moment transferred to link k around its center of mass due to the rotational effect of frame j is given by:

$$\boldsymbol{\mu}_{Ckj} = \boldsymbol{\mu}^{\tau}_{Ckj} + \boldsymbol{\mu}^{n}_{Ckj} + \boldsymbol{\mu}^{cor}_{Ckj} \tag{19}$$

IV. Calculating Christoffel symbols

For articulated rigid bodies in a weightless environment (no gravitational field) equation (3) becomes:

$$\sum_j a_{kj}\ddot{q}_j + \sum_{i,j} c^k_{ji}\dot{q}_j\dot{q}_i = \tau_k \tag{20}$$

We propose a situation where joints i and j of the articulated rigid bodies are in motion with constant angular velocities, while the other joints are fixed, then the left hand side of equation (20) becomes:

$$\sum_{i,j} c^k_{ji}\dot{q}_j\dot{q}_i = c^k_{jj}\dot{q}^2_j + 2c^k_{ji}\dot{q}_j\dot{q}_i + c^k_{ii}\dot{q}^2_i \tag{21}$$

Considering the frame injection principal, the right hand side of equation (20) is the torque resulting from the sum of three inertial moments:

$$\tau_k = \tau_{k_j} + \tau_{k_{ji}} + \tau_{k_i} \tag{22}$$

Where:

- τ_{k_j} is the the torque due to the Centrifugal effect resulting from motion of joint j.
- $\tau_{k_{ji}}$ is the the torque due to the Coriolis effect resulting from motion of joints j and i.
- τ_{k_i} is the the torque due to the Centrifugal effect resulting from motion of joint i.

From equations (21) and (22) we find that:

$$\tau_{k_j} = c^k_{jj}\dot{q}^2_j \tag{23}$$

$$\tau_{k_{ji}} = 2c^k_{ji}\dot{q}_j\dot{q}_i \tag{24}$$

$$\tau_{k_i} = c^k_{ii}\dot{q}^2_i \tag{25}$$

In such a case, to calculate c^k_{jj}, c^k_{ji}, and c^k_{ii}. We can assign a unitary value to the angular velocities $\dot{q}_j$ and $\dot{q}_i$, then equations (23,24,25) are interpreted as:

- The Christoffel symbol c^k_{ji} is equal to half the torque $\tau_{k_{ji}}$, which acts on joint k due to Coriolis effect resulting from the unit angular velocities at joints j and i.
- The Christoffel symbol c^k_{jj} is equal to the torque τ_{k_j}, which acts on joint k due to Centrifugal effect resulting from the unit angular velocity at joint j.
- The same applies for Christoffel symbol c^k_{ii}, which results from c^k_{jj} after a change of index.

To calculate the Christoffel symbols c^k_{ji} we apply backward recursion on the forces and moments shown in Figure 3, where:

- $\boldsymbol{h}^k_{ji}$: is half the inertial moment, $\boldsymbol{\mu}^{cor}_{Ckj}$, at the center of mass of link k. It is due to Coriolis effect resulting from a unit angular velocity at joints j and i . From equation (17) of the frame injection principal, $\boldsymbol{h}^k_{ji}$ is given by:

$$\boldsymbol{h}^k_{ji} = \frac{1}{2}\boldsymbol{\mu}^{cor}_{Ckj} = \frac{1}{2}(\mathbf{L}_k\boldsymbol{k}_j)\times\boldsymbol{k}_i \tag{26}$$

- $\boldsymbol{f}^k_{ji}$: is half the inertial force at the center of mass of link k due to Γ^{cor}_{Ckj}. Or the Coriolis effect for a unit angular velocity at joints j and i. From equation (11) of the frame injection principal, $\boldsymbol{f}^k_{ji}$ is given by:

$$\boldsymbol{f}^k_{ji} = \frac{1}{2}m_k\Gamma^{cor}_{Ckj} = m_k\boldsymbol{k}_j\times(\boldsymbol{k}_i\times\boldsymbol{p}_{Cki}) \tag{27}$$

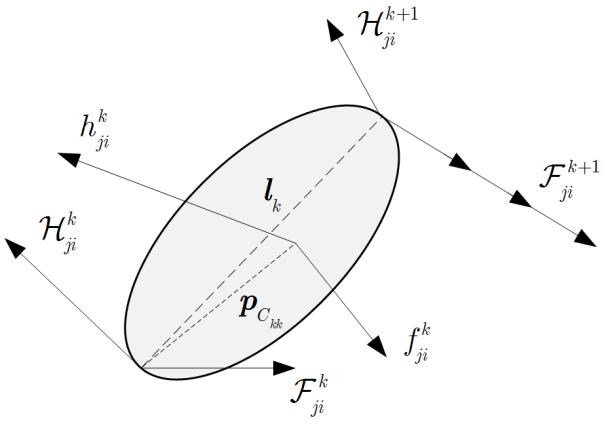

Fig. 3
Backward recursion on moments and forces

- We calculate the Christoffel symbols c^k_{ji} recursively, by applying a backward recursion on Figure 3 for: 1) the inertial forces $\boldsymbol{f}^k_{ji}$ and 2) the inertial moments $\boldsymbol{h}^k_{ji}$:

$$\boldsymbol{\mathcal{F}}^k_{ji} = \boldsymbol{\mathcal{F}}^{k+1}_{ji} + \boldsymbol{f}^k_{ji} \tag{28}$$

$$\boldsymbol{\mathcal{H}}^k_{ji} = \boldsymbol{\mathcal{H}}^{k+1}_{ji} + \boldsymbol{h}^k_{ji} + \boldsymbol{p}_{Ckk}\times\boldsymbol{f}^k_{ji} + \boldsymbol{l}_k\times\boldsymbol{\mathcal{F}}^k_{ji} \tag{29}$$

$$c^k_{ji} = \boldsymbol{k}^{\mathrm{T}}_k\boldsymbol{\mathcal{H}}^k_{ji} \tag{30}$$

Where $\boldsymbol{\mathcal{F}}^k_{ji}$ is half the inertial force calculated recursively at joint k due to the unit angular velocity at joints j and i. $\boldsymbol{\mathcal{H}}^k_{ji}$ is half the inertial moment calculated recursively at joint k due to the unit angular velocity at joints j and i, and the superscript T in $\boldsymbol{k}^{\mathrm{T}}_k$ is to denote the transpose. By applying a similar approach on normal accelerations we can calculate c^k_{ii} (c^k_{jj}). It is noticed that the resulting equations for calculating c^k_{ii} and c^k_{jj} is exactly similar to the ones in the presented algorithm (for calculating c^k_{ji}) only with indices changed.

V. Implementation and results

To prove the validity of the proposed method for calculating Christoffel symbols, and to assess its performance, a comparison with symbolic Lagrangian based method was performed, the code used is provided in the public repository give in [17]. The robot used to run the test is 5 degrees of freedom serially linked robot, its structure is described in the file (robotStructure_5DOF.mat), this robot is generated using the file (generateRandomRobot.m) found in folder (Generating symbolic equations of Christoffel using Lagrangian) of [17] in which the mass of each link was generated randomly in the range [0,1] kg. The inertial tensor of each link was generated as random positive definite matrix in which each

TABLE I

COMPARISON FOR CALCULATING CHRISTOFFEL OF 5DOF SERIALLY LINKED ROBOT USING DIFFERENT METHODS

Criteria	Lagrangian (Optimized)	Lagrangian (Not optimized)	Proposed
Size of generated file (bytes)	778 732 bytes	67 873 609 bytes	4 497 bytes
Off-line time for function generation	8 days	897 sec	-
On-line execution time (seconds)	4.6e-04	96 sec	9.9e-5

TABLE II

COMPUTATIONAL COMPLEXITY OF THE PROPOSED METHOD

Additions	Multiplications
$12n^3 + 19n^2 + 40n + 1$	$\frac{21}{2}n^3 + \frac{45}{2}n^2 + 49n$

element of the matrix is in the range [0,1] kg.m^2. Denavit-Hartenberg (DH) parameters of each link were generated randomly. Afterwards, using MATLAB, Christoffel symbols of the robot were calculated using:

1) The proposed algorithm, which is implemented in the MATALB function (christoffelNumerically.m).
2) An off-line generated MATLAB function (christoffel_symbolicallyGenerated5DOF.m) which contains the symbolic equations generated using Lagrangian method. In this case, the optimization option of the code generator was set to true, as to optimize the generated symbolic equations.
3) Using an off-line generated MATLAB function (christoffel_symbolicallyGenerated5DOFnoOptimization.m) which contains the symbolic equations generated using Lagrangian method. In this case, the optimization option of the symbolic equation generator was set to false.

Afterwards, the Christoffel symbols of the manipulator were calculated twice, once using symbolic function and another using the proposed method. Table I shows a comparison of achieved results, the proposed recursive method is superior in various aspects, including, in terms of execution time (4.6X times faster for a 5DOF robot). For a 6DOF robot the script has been running for two months without finishing the symbolic equations generation.

Table II shows a summary of the computational complexity of the proposed algorithm measured in the number of floating point operations (additions and multiplications) as function of n, the number of DOF of the robot. A detailed breakdown of the computational complexity is found in the excel file (Operation_Count.ods) found in [17].

Finally, to measure the numerical accuracy of the calculations, the following metric-value was defined:

$$e = \frac{2}{n^3} \sum_{i,j,k} \left| \frac{c^k_{ji} - \hat{c}^k_{ji}}{c^k_{ji} + \hat{c}^k_{ji}} \right| \quad for\, each, c^k_{ji} \neq 0 \qquad (31)$$

Where e is the relative error, c^k_{ji} is the Christoffel symbols calculated using proposed method and $\hat{c}^k_{ji}$ is the Christoffel symbols calculated using symbolic method. From various calculations using randomly generated configurations, the maximum error achieved is $2.196e-14$.

VI. CONCLUSION

In this study we proposed recursive algorithm for calculating Christoffel symbols efficiently for serially linked robots. The algorithm achieves better efficiency over Lagrangian based symbolic method. This increase in efficiency is achieved by performing backward recursion on forces and moments. As compared to symbolic method, computational testing proves that the proposed algorithm is (1) efficient (faster execution time), (2) precise (negligible numerical error), and most importantly (3) it does not require a time consuming off-line code generation phase.

REFERENCES

[1] Roy Featherstone and David Orin. Robot dynamics: equations and algorithms. In *ICRA*, pages 826–834, 2000.

[2] CA Balafoutis. A survey of efficient computational methods for manipulator inverse dynamics. *Journal of Intelligent and Robotic Systems*, 9(1-2):45–71, 1994.

[3] Oussama Khatib. Dynamic control of manipulator in operational space. In *Proc. 6th IFToMM World Congress on Theory of Machines and Mechanisms*, pages 1128–1131, 1983.

[4] Oussama Khatib. A unified approach for motion and force control of robot manipulators: The operational space formulation. *Robotics and Automation, IEEE Journal of*, 3(1):43–53, 1987.

[5] Kyong-Sok Chang. *Efficient algorithms for articulated branching mechanisms: dynamic modeling, control, and simulation*. PhD thesis, Citeseer, 2000.

[6] Kyong-Sok Chang and Oussama Khatib. Efficient recursive algorithm for the operational space inertia matrix of branching mechanisms. *Advanced Robotics*, 14(8):703–715, 2001.

[7] Lorenzo Sciavicco, Bruno Siciliano, and Luigi Villani. Lagrange and newton-euler dynamic modeling of a gear-driven robot manipulator with inclusion of motor inertia effects. *Advanced robotics*, 10(3):317–334, 1995.

[8] Yang Liu and Hongnian Yu. A survey of underactuated mechanical systems. *IET Control Theory & Applications*, 7(7):921–935, 2013.

[9] Charles P Neuman and John J Murray. Symbolically efficient formulations for computational robot dynamics. *Journal of robotic systems*, 4(6):743–769, 1987.

[10] Wisama Khalil. Dynamic modeling of robots using recursive newton-euler techniques. In *ICINCO2010*, 2010.

[11] Herman Hoifodt. Dynamic modeling and simulation of robot manipulators: the newton-euler formulation. 2011.

[12] John YS Luh, Michael W Walker, and Richard PC Paul. On-line computational scheme for mechanical manipulators. *Journal of Dynamic Systems, Measurement, and Control*, 102(2):69–76, 1980.

[13] Roy Featherstone. A divide-and-conquer articulated-body algorithm for parallel o (log (n)) calculation of rigid-body dynamics. part 1: Basic algorithm. *The International Journal of Robotics Research*, 18(9):867–875, 1999.

[14] Roy Featherstone. A divide-and-conquer articulated-body algorithm for parallel o (log (n)) calculation of rigid-body dynamics. part 2: Trees, loops, and accuracy. *The International Journal of Robotics Research*, 18(9):876–892, 1999.

[15] Roy Featherstone. Efficient factorization of the joint-space inertia matrix for branched kinematic trees. *The International Journal of Robotics Research*, 24(6):487–500, 2005.

[16] M. Safeea, R. Bearee, and P. Neto. Reducing the computational complexity of mass-matrix calculation for high dof robots. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5614–5619, Oct 2018.

[17] Code for calculating christoffel symbols recursively. `https://github.com/Modi1987/recursiveChristoffelSymbols`. Accessed: 2019-02-01.

[18] R Gunawardana and F Ghorbel. On the uniform boundedness of the coriolis/centrifugal terms in the robot equations of motion. *International Journal of Robotics and Automation*, 14(2):45–53, 1999.

[19] Rafael Kelly. Comments on" adaptive pd controller for robot manipulators. *IEEE Transactions on Robotics and Automation*, 9(1):117–119, 1993.

[20] Patrizio Tomei. Adaptive pd controller for robot manipulators. *IEEE Transactions on Robotics and Automation*, 7(4):565–570, 1991.

[21] Jose Alvarez-Ramirez, Ilse Cervantes, and Rafael Kelly. Pid regulation of robot manipulators: stability and performance. *Systems & control letters*, 41(2):73–83, 2000.

[22] Ilse Cervantes and Jose Alvarez-Ramirez. On the pid tracking control of robot manipulators. *Systems & control letters*, 42(1):37–46, 2001.

[23] Zhihua Qu and John Dorsey. Robust tracking control of robots by a linear feedback law. *IEEE Transactions on Automatic Control*, 36(9):1081–1084, 1991.

[24] R Ortega, A Loria, and R Kelly. A semiglobally stable output feedback pid regulator for robot manipulator, automatic control. *IEEE Transaction August*, 40, 1995.

[25] Juan Ignacio Mulero-Martinez. An improved dynamic neurocontroller based on christoffel symbols. *IEEE transactions on neural networks*, 18(3):865–879, 2007.

[26] Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, and Giuseppe Oriolo. *Robotics: modelling, planning and control*. Springer Science & Business Media, 2010.

[27] Thomas Lipp and Stephen Boyd. Minimum-time speed optimisation over a fixed path. *International Journal of Control*, 87(6):1297–1311, 2014.

[28] S Yin and J Yuh. An efficient algorithm for automatic generation of manipulator dynamic equations. In *Robotics and Automation, 1989. Proceedings., 1989 IEEE International Conference on*, pages 1812–1817. IEEE, 1989.

[29] Kevin M. Lynch and Frank C. Park. *Modern Robotics: Mechanics, Planning, and Control*. Cambridge University Press, New York, NY, USA, 1st edition, 2017.

[30] Steven M LaValle. *Planning algorithms*. Cambridge university press, 2006.

[31] Bruno Siciliano and Oussama Khatib. *Springer handbook of robotics*. Springer, 2016.