Robot Dynamics: a Recursive Algorithm for Efficient Calculation of Christoffel Symbols

Mohammad Safeea[a,b], Pedro Neto[a], Richard Bearee[b]

[a] *University of Coimbra, Department of Mechanical Engineering, 3030-788 Coimbra, Portugal*

[b] *Arts et Métiers, LISPEN, 59800 Lille, France*

Abstract

Christoffel symbols of the first kind are very important in robot dynamics. 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 the literature of robot dynamics, Christoffel symbols of the first kind are calculated from Lagrangian dynamics using an off-line generated symbolic formula. In this study we present a novel and efficient recursive, non-symbolic, method where Christoffel symbols of the first kind are calculated on-the-fly based on the inertial parameters of robot's links and their transformation matrices. 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.

Keywords: Dynamics, Christoffel symbols, recursive algorithms

Email address: ms@uc.pt, pedro.neto@dem.uc.pt, Richard.BEAREE@ensam.eu (Mohammad Safeea[a,b], Pedro Neto[a], Richard Bearee[b])

Nomenclature	
$\mathcal{L}$	Lagrangian function of kinematic chain (robot)
$\mathcal{T}$	kinetic energy
$\mathcal{U}$	potential energy
$\boldsymbol{\tau}$	vector of joints torques of the robot
τ_k	torque at joint k of the robot
$q_j, \dot{q}_j, \ddot{q}_j$	angular position, velocity and acceleration of joint j of the robot
g_k	torque due to gravity at joint k of the robot
a_{kj}	element (k, j) of the mass matrix of the robot
c^k_{ji}	Christoffel symbol of the first kind
$\boldsymbol{p}_{Ckj}$	vector connecting the origin of frame j and the center of mass of link k
$\boldsymbol{k}_j$	unit vector associated with the z axis of joint j
$\boldsymbol{\mu}_{Ckj}, \boldsymbol{\Gamma}_{Ckj}$	inertial moment and linear acceleration of center of mass of link k transferred by frame j
$\boldsymbol{\Gamma}^{\tau}_{Ckj}, \boldsymbol{\Gamma}^{n}_{Ckj}, \boldsymbol{\Gamma}^{cor}_{Ckj}$	tangential, normal and Coriolis acceleration vectors transferred by frame j into the center of mass of link k
$\boldsymbol{\mu}^{\tau}_{Ckj}, \boldsymbol{\mu}^{n}_{Ckj}, \boldsymbol{\mu}^{cor}_{Ckj}$	inertial moment vectors due to angular acceleration, centrifugal and Coriolis effects transferred by frame j into the center of mass of link k
$\mathbf{R}_k$	rotation matrix of link k with respect to base frame of the robot
$\mathbf{I}^k_k$	inertial tensor of link k taken at its center of mass with respect to the link's local frame
$\boldsymbol{h}^k_{ji}, \boldsymbol{f}^k_{ji}$	half of the inertial force/moment at the center of mass of link k due to Coriolis effect resulting from a unit angular velocity at joints j and i
$\boldsymbol{\mathcal{F}}^k_{ji}, \boldsymbol{\mathcal{H}}^k_{ji}$	half of the inertial force/moment calculated recursively at the joint k due to Coriolis effect resulting from a unit angular velocity at joints j and i

1. Introduction

Christoffel symbols are important tools in applied sciences, engineering, mathematics and physics. In the latter they appear in rigid body dynamics [1] and general relativity [2]. In the area of robotics, Christoffel symbols of the first kind appear when deducing the equation of robot dynamics using the Lagrangian:

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

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 $\boldsymbol{\tau}$:

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

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} represents Christoffel symbols of the first kind (later in the document referred to by Christoffel symbols). From [3], c^k_{ji} is given by:

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

The Lagrangian method is a straight forward approach for deducing the dynamics equations of mechanical systems. However, the method requires partial differentiation, which makes it unpractical to apply manually for complex systems. Thus, symbolic manipulation methods have been utilized to perform the differentiation by a computer [4]. Nevertheless, the resulting equations generated by a computer lack the efficiency in terms of execution-time, this fact becomes more noticeable for robots with highe number of joints or Degrees Of Freedom (DOF) as noted in [5] and most remarkably in [6], where the author compares the execution-times required to run dynamics simulations based on models derived by Newton-Euler recursive technique and Euler-Lagrange technique. It is reported execution-times difference of order of magnitude which put the case in favor of the Newton-Euler recursion method.

Owing to their efficiency, researchers developed various recursive algorithms [7], namely for calculating the inverse dynamics [8], the forward dynamics [9, 10] and the joint space inertia matrix [11, 12]. Nevertheless, we are not aware of any recursive algorithm 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, are available in the supplementary material.

The article is organized as the following: Section 2 discusses the motivation and the contributions of the proposed study. Section 3 lists the principles used in this study for describing the dynamics of serially-linked articulated rigid bodies. In Section 4 the proposed algorithm for calculating Christoffel symbols recursively is deduced. In Section 5 tests and results are presented, where the proposed recursive algorithm is evaluated against Lagrangian method for calculating Christoffel symbols. Section 6 describes an example showing the benefits of the proposed algorithm as applied to industrial robot manipulators. Finally the article ends with the conclusion in Section 7.

2. Motivation and Contribution

Calculating Christoffel symbols of the first kind is very important in robot dynamics. Hence, Christoffel symbols have been used for solving various robotics problems. In [13] they are used for calculating the bounds on the Coriolis/Centrifugal matrix. These bounds play an important role for designing and tuning various robot controllers [14, 15, 16, 17, 18, 19]. In addition, Christoffel symbols have been used in a dynamic neurocontroller of robotic arms [20]. They can also be used to calculate a special form of Coriolis matrix that preserves the skew symmetry property [21] (an essential property for various control algorithms). Christoffel symbols are also important for planning time optimal trajectories [22] and optimal velocity-profile generation [23, 24]. In such a case, the geometric path is parametrized using a vector function of a scalar parameter $\theta(t)$. Consequently, 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 [23]:

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

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 [25] 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 [26, 21, 27], including the Handbook of Robotics [28] in page 44, where the deduction of Christoffel symbols is introduced in the Dynamics chapter under the subsection "2.3.2 Lagrange Formulation". Nevertheless, symbolic methods for performing the calculations have major drawbacks, namely:

- The symbolic manipulation of the equations is time consuming, so it has to be performed off-line;
- For high DOF, the symbolic equations become very complex resulting in much slower execution times than recursive methods, a fact reported in literature [5].

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

- Unlike the symbolic methods, recursive methods can be used on-the-fly, and do 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 the end-effector (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.

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.

3. Theory and principles

The proposed algorithm builds on what we call the frame injection principle introduced in [12], also illustrated in Figure 1, where a frame j attached to joint j (according to the modified Denavit Hartenberg convention [29]) transfers 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 axis of rotation, or the z axis of frame j. 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 subscript C is used to refer to the center of mass of link k. The same subscript notation will hold throughout this study for denoting frame-link interaction of cause-and-effect unless stated otherwise.

3.1. 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:

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

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 and 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 joint 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:

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

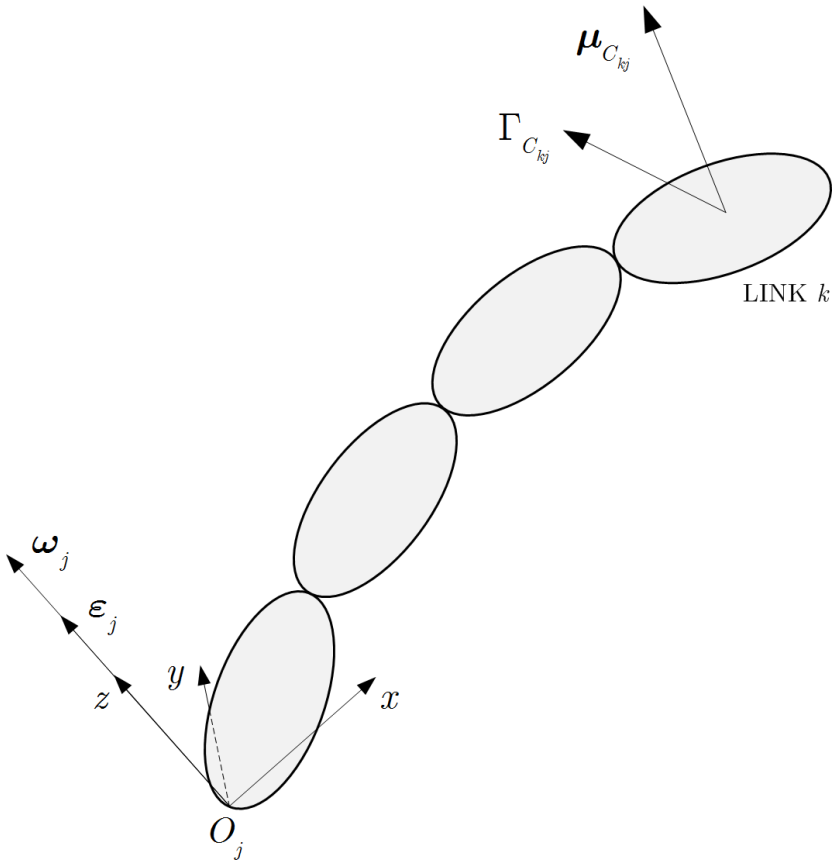

Figure 1: Inertial moment $\boldsymbol{\mu}_{Ckj}$ and linear acceleration Γ_{Ckj} of center of mass of link k transferred by frame j

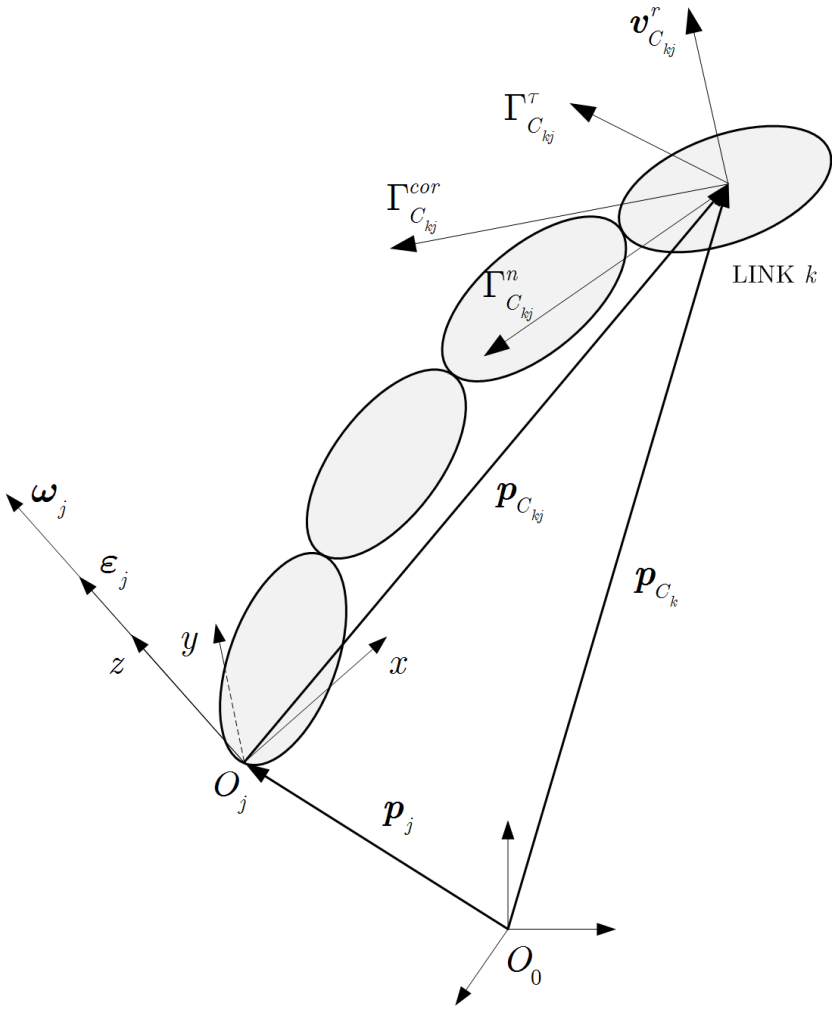

Figure 2: Tangential, normal and Coriolis accelerations of center of mass of link k transferred by frame j

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 $\boldsymbol{v}^r_{Ckj}$ is the velocity transferred to the center of mass of link k from frames $j+1$ up to frame k. The superscript r is used to denote that this is a relative velocity and C to refer to the center of mass 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:

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

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

Each frame j transfers to link k three inertial moments, the first of which is due to angular acceleration of frame j:

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

where $\boldsymbol{\mu}^{\tau}_{Ckj}$ 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:

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

where $\boldsymbol{\omega}_{kj}^{r}$ can be calculated from:

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

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:

$$\boldsymbol{\mu}_{\mathrm{C}kj} = \boldsymbol{\mu}_{\mathrm{C}kj}^{\tau} + \boldsymbol{\mu}_{\mathrm{C}kj}^{n} + \boldsymbol{\mu}_{\mathrm{C}kj}^{cor} \tag{19}$$

4. Calculating Christoffel symbols

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

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

We propose a scenario where only 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:

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

Considering the frame injection principle, 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. Thus, it is a function of $\dot{q}_j^2$;
- $\tau_{k_{ji}}$ is the the torque due to the Coriolis effect resulting from motion of joints j and i. Thus, it is a function of the product $\dot{q}_j\dot{q}_i$;
- τ_{k_i} is the the torque due to the Centrifugal effect resulting from motion of joint i. Thus, it is a function of $\dot{q}_i^2$.

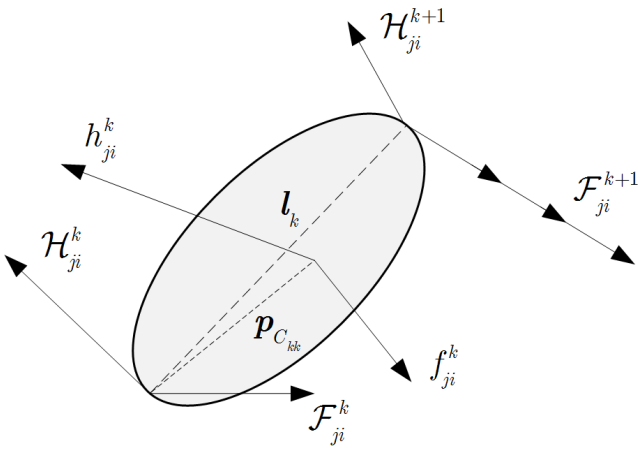

Figure 3: Backward recursion on moments and forces

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

$$\tau_{k_j} = c_{jj}^{k}\dot{q}_j^2 \tag{23}$$

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

$$\tau_{k_i} = c_{ii}^{k}\dot{q}_i^2 \tag{25}$$

In such a case, to calculate c_{jj}^{k}, c_{ji}^{k} and c_{ii}^{k} we 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_{ji}^{k} is equal to half of 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_{jj}^{k} 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_{ii}^{k} which results from c_{jj}^{k} after a change of index.

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

- $\boldsymbol{h}_{ji}^{k}$ is half of the inertial moment $\boldsymbol{\mu}_{Ckj}^{cor}$ 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 principle, $\boldsymbol{h}_{ji}^{k}$ is given by:

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

Table 1: Comparison for calculating Christoffel symbols of a 5 DOF serially linked robot using different methods

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

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

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

We calculate the Christoffel symbols c_{ji}^{k} recursively, by applying a backward recursion on Figure 3 for the inertial forces $\boldsymbol{f}_{ji}^{k}$ and the inertial moments $\boldsymbol{h}_{ji}^{k}$:

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

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

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

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

5. 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 is provided in the supplementary material. The robot used to run the test is a 5 DOF serially linked robot, its structure is described in the file `robotStructure_5DOF.mat`. This robot is generated using the file `generateRandomRobot.m` 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 element of the matrix is in the range [0,1] $\mathrm{kg\,m^2}$. Denavit-Hartenberg (DH) parameters of each

link were also 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 which contains the symbolic equations generated using Lagrangian method, `chri_symbGen5DOF.m`. 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 which contains the symbolic equations generated using Lagrangian method `chri_symbGen5DOFnoOpt.m`. In this case, the optimization option of the symbolic equation generator was set to false.

The Christoffel symbols of the manipulator were calculated twice, once using symbolic function and another using the proposed method. Table 1 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 5 DOF robot). The tests were carried out on a personal computer with Intel(R) Core(TM) i7-6850K CPU @ 3.6 GHz, under Windows 10, running MATLAB 2018a. For a 6 DOF robot, the script has been running for two months without finishing the symbolic equations generation (the automatic optimization of the generated equations is extremely time consuming). On the other hand, using the proposed algorithm to calculate Christoffel symbols for 6 DOF robot requires 13.3e-5 seconds, the file with the test is `a01_timeExecution6DOF.m` found in the supplementary material.

Table 2 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 file `Operation_Count.ods` found in the supplementary material.

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

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

where e is the relative error, c^k_{ji} is Christoffel symbol calculated using the proposed method and $\hat{c}^k_{ji}$ is Christoffel symbol calculated using the symbolic method. From various calculations using randomly generated configurations, the maximum (worst) e value achieved is $2.196e-14$, indicating that the error is so small mainly due to numerical rounding errors.

6. Application Example

An important application for the proposed algorithm is in minimum-time trajectory optimization for industrial manipulators working in flexible manufac-

Table 2: Computational complexity of the proposed method

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

turing. In such a case, calculating Christoffel symbols efficiently and on-the-fly is of importance. Because, they enter into the formulation of the minimum-time optimization problem based on robot dynamics as shown in [30, 31], where the elements of vector $\tilde{\boldsymbol{c}}(\boldsymbol{q})$ in equation (5) are calculated based on the mass matrix and Christoffel symbols:

$$\tilde{c}_k = \sum_j a_{kj} q''_j + \sum c^k_{ij} q'_i q'_j \tag{32}$$

where $\tilde{c}_k$ is the k^{th} element of the vector $\tilde{\boldsymbol{c}}(\boldsymbol{q})$, q'_j is calculated from:

$$q'_j = \frac{dq_j}{d\theta} \tag{33}$$

and q''_j is calculated from:

$$q''_j = \frac{d^2 q_j}{d\theta^2} \tag{34}$$

A representative scenario is shown in Figure 4, where a 7 DOF industrial robot is used to palletize objects in a flexible-manufacturing production line. Optimizing the robot motion for achieving minimum-time trajectory is very important for achieving high productivity. However, due to the flexible-manufacturing requirements, the robot is required to manipulate various types of objects, with different inertial data (inertial tensor, mass, center of mass). Consequently, the inertial data of the object has to be taken into consideration for optimizing the robot motion while moving the object. This can be done by considering the last link of the robot and the object as a one body when the robot is manipulating it on the planned path.

In such a case, a laser sensor is used to read the bar-code sticker (on the object) which includes the inertial data of the object. When a new object (with different inertial data) is present, the control algorithm calculates the equivalent inertial data of the last link coupled with the object. Considering both, the inertia of the last link of the robot and the object. Afterwards, the optimization problem is invoked, where Christoffel symbols are calculated efficiently and on-the-fly using our algorithm. In comparison, deducing the equations of Christoffel symbols by symbolic manipulation creates a bottleneck in the problem formulation, where generating the symbolic equations is (extremely) time consuming (requires an off-line generation phase which is eliminated by our recursive algorithm). This limits the applicability of the traditional method for

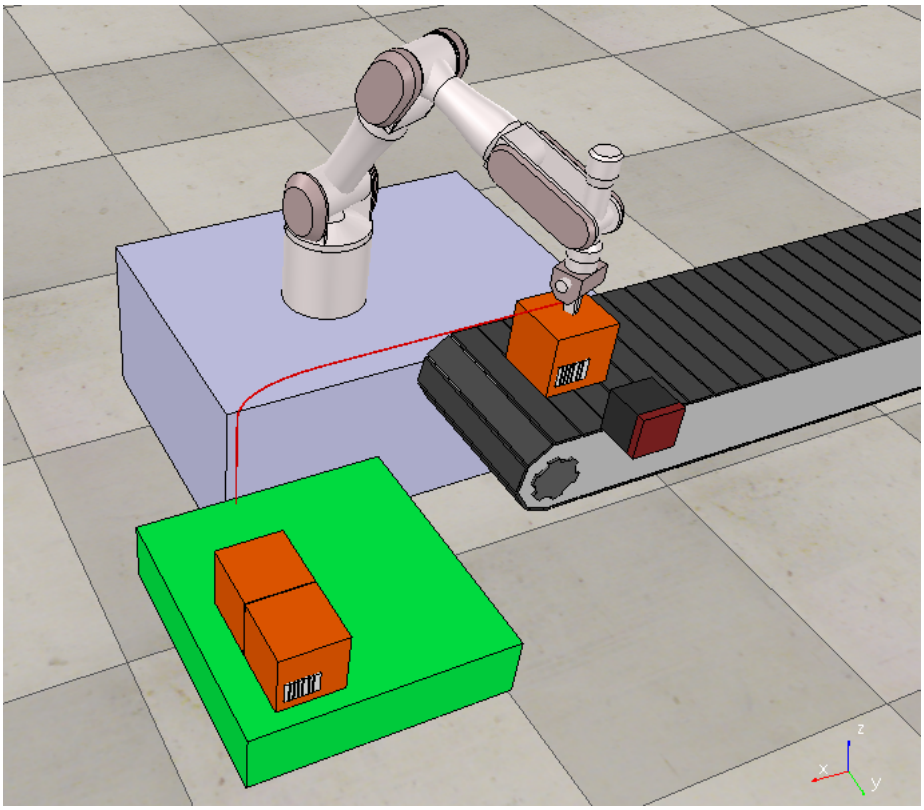

Figure 4: Minimum-time trajectory optimization for industrial manipulator performing palatalization in flexible-manufacturing scenario.

calculating Christoffel symbols for variable inertias in time critical operations. After formulating the problem, the time required to perform the optimization is of order of seconds [31].

This application example shows the importance of our algorithm for performing Christoffel symbols calculation in a practical application, where our algorithm offers two fundamental advantages:

1. Christoffel symbols are calculated on-the-fly (without requiring an extremely time consuming off-line phase), even when the inertial data are changing;
2. Christoffel symbols are calculated more efficiently using our recursive algorithm than using the traditional method (symbolic equation generation).

7. 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.

8. Acknowledgements

This research was partially supported by Portugal 2020 project DM4Manufacturing POCI-01-0145-FEDER-016418 by UE/FEDER through the program COMPETE 2020, and the Portuguese Foundation for Science and Technology (FCT) SFRH/BD/131091/2017 and COBOTIS (PTDC/EMEEME/ 32595/2017).

References

[1] L. Sciavicco, B. Siciliano, L. Villani, Lagrange and newton-euler dynamic modeling of a gear-driven robot manipulator with inclusion of motor inertia effects, Advanced robotics 10 (1995) 317–334.

[2] C. W. Misner, K. S. Thorne, J. A. Wheeler, Gravitation, W.H. Freeman and Company, 1973.

[3] Y. Liu, H. Yu, A survey of underactuated mechanical systems, IET Control Theory & Applications 7 (2013) 921–935.

[4] C. P. Neuman, J. J. Murray, Symbolically efficient formulations for computational robot dynamics, Journal of robotic systems 4 (1987) 743–769.

[5] W. Khalil, Dynamic modeling of robots using recursive newton-euler techniques, in: ICINCO2010, 2010.

[6] H. Hoifodt, Dynamic modeling and simulation of robot manipulators: the newton-euler formulation (2011).

[7] R. Featherstone, D. Orin, Robot dynamics: equations and algorithms, in: ICRA, 2000, pp. 826–834.

[8] J. Y. Luh, M. W. Walker, R. P. Paul, On-line computational scheme for mechanical manipulators, Journal of Dynamic Systems, Measurement, and Control 102 (1980) 69–76.

[9] R. 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 (1999) 867–875.

[10] R. 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 (1999) 876–892.

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

[12] M. Safeea, R. Bearee, 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), 2018, pp. 5614–5619. doi:10.1109/IROS.2018.8593775.

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

[14] R. Kelly, Comments on" adaptive pd controller for robot manipulators, IEEE Transactions on Robotics and Automation 9 (1993) 117–119.

[15] P. Tomei, Adaptive pd controller for robot manipulators, IEEE Transactions on Robotics and Automation 7 (1991) 565–570.

[16] J. Alvarez-Ramirez, I. Cervantes, R. Kelly, Pid regulation of robot manipulators: stability and performance, Systems & control letters 41 (2000) 73–83.

[17] I. Cervantes, J. Alvarez-Ramirez, On the pid tracking control of robot manipulators, Systems & control letters 42 (2001) 37–46.

[18] Z. Qu, J. Dorsey, Robust tracking control of robots by a linear feedback law, IEEE Transactions on Automatic Control 36 (1991) 1081–1084.

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

[20] J. I. Mulero-Martinez, An improved dynamic neurocontroller based on christoffel symbols, IEEE transactions on neural networks 18 (2007) 865–879.

[21] B. Siciliano, L. Sciavicco, L. Villani, G. Oriolo, Robotics: modelling, planning and control, Springer Science & Business Media, 2010.

[22] F. Pfeiffer, R. Johanni, A concept for manipulator trajectory planning, IEEE Journal on Robotics and Automation 3 (1987) 115–123.

[23] T. Lipp, S. Boyd, Minimum-time speed optimisation over a fixed path, International Journal of Control 87 (2014) 1297–1311.

[24] Á. Nagy, I. Vajk, Non-convex time-optimal trajectory planning for robot manipulators, Journal of Dynamic Systems, Measurement, and Control (2019).

[25] S. Yin, J. Yuh, An efficient algorithm for automatic generation of manipulator dynamic equations, in: Robotics and Automation, 1989. Proceedings., 1989 IEEE International Conference on, IEEE, 1989, pp. 1812–1817.

[26] K. M. Lynch, F. C. Park, Modern Robotics: Mechanics, Planning, and Control, 1st ed., Cambridge University Press, New York, NY, USA, 2017.

[27] S. M. LaValle, Planning algorithms, Cambridge university press, 2006.

[28] B. Siciliano, O. Khatib, Springer handbook of robotics, Springer, 2016.

[29] J. J. Craig, Introduction to robotics: mechanics and control, volume 3, Pearson Prentice Hall Upper Saddle River, 2005.

[30] P. Reynoso-Mora, W. Chen, M. Tomizuka, A convex relaxation for the time-optimal trajectory planning of robotic manipulators along predetermined geometric paths, Optimal Control Applications and Methods 37 (2016) 1263–1281.

[31] D. Verscheure, B. Demeulenäre, J. Swevers, J. De Schutter, M. Diehl, Practical time-optimal trajectory planning for robots: a convex optimization approach, IEEE Transactions on Automatic Control (2008).