Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 55-37 (2022) 74–80

Application of Pseudo-Symbolic Dynamic Modeling (PSDM) in the Modeling & Calibration of a 6-DOF Articulated Robot

Steffan Lloyd * **Rishad Irani** * **Mojtaba Ahmadi** *

* *Carleton University, Ottawa, ON, K1S 5B6, Canada*
(e-mails: {steffan.lloyd, rishad.irani, mojtaba.ahmadi}@carleton.ca)

Abstract: This paper describes the modeling, calibration, and benchmarking of the pseudo-symbolic dynamic modeling (PSDM) method on a Denso VS-6556G manipulator outfitted for robotic deburring purposes. PSDM is a numerical dynamic modeling method that allows for fast real-time evaluation and returns the dynamic model in regressive form, allowing for straightforward model calibration. An overview of the PSDM method is given, and the model derivation for the Denso manipulator is demonstrated. This model is calibrated experimentally, along with a joint actuator model and LuGre friction model for each joint. The calibration is highly accurate, capturing 98–100% of the motor, gravity, and friction effects, while the high-speed acceleration fitting captures 81–99% of the observed torque effects. Benchmarking of the PSDM model shows it to be significantly faster than the reverse Newton-Euler algorithm.

Keywords: Nonlinear system identification, software for system identification, identification and control methods, robotic manipulators

1. INTRODUCTION

Accurate dynamic modeling of robotic manipulators is essential for control system design, simulation, motion planning, state estimation, manipulator design, and much more. Classically, the dynamic model of an n-link manipulator is represented with the differential equation

$$\boldsymbol{\tau} = \mathbf{D}(\mathbf{q})\,\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\,\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q}), \tag{1}$$

where $\mathbf{q}$ is the generalized joint coordinates, $\mathbf{D}(\mathbf{q})$ is the *mass matrix*, $\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\,\dot{\mathbf{q}}$ is a vector of Coriolis and centrifugal terms, and $\mathbf{G}(\mathbf{q})$ is the *gravity vector*. Evaluation of this model is a function of the $10n$ parameters of masses, centers of gravity, and moments of inertia of each linkage,

$$\mathbf{X} = \begin{bmatrix} m_1 & r_{x1} & r_{y1} & r_{z1} & I_{xx1} & I_{yy1} & I_{zz1} & I_{xy1} & I_{xz1} & I_{yz1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ m_n & r_{xn} & r_{yn} & r_{zn} & I_{xxn} & I_{yyn} & I_{zzn} & I_{xyn} & I_{xzn} & I_{yzn} \end{bmatrix}. \tag{2}$$

Typically, these inertial parameters are not known, and as such, the dynamic model cannot be leveraged accurately in any useful manner. A key result in the dynamic modeling of manipulators, however, is that (1) is known to be *linear* with respect to a set of ℓ base inertial parameters $\boldsymbol{\theta}_b$, and as such, it is possible to rearrange into *regressor form*,

$$\boldsymbol{\tau} = \mathbf{Y}(\mathbf{q},\dot{\mathbf{q}},\ddot{\mathbf{q}})\,\boldsymbol{\theta}_b, \tag{3}$$

where $\mathbf{Y}(\mathbf{q},\dot{\mathbf{q}},\ddot{\mathbf{q}})$ is the $n \times \ell$ *manipulator regressor*. The inertial parameter vector $\boldsymbol{\theta}_b$ contains all the information minimally required to calibrate the dynamic model; however, due to constrained motion between joints, will have fewer entries than $\mathbf{X}$, such that $\ell \leq 10n$. In this form, with experimental knowledge of $\boldsymbol{\tau}$, $\mathbf{q}$, $\dot{\mathbf{q}}$ and $\ddot{\mathbf{q}}$, one can use linear system identification techniques to identify $\boldsymbol{\theta}_b$ and calibrate the dynamic model.

Many methods have been proposed to compute the dynamic model efficiently. Lagrangian methods (Uicker, 1969) return the model symbolically; however, they scale extremely poorly to higher degrees of freedom (DOF), with complexity $O(n^4)$. Moreover, the model is returned in canonical form as in (1), but rearrangement into regressor form is nontrivial. Perhaps the most common method is the *recursive Newton-Euler* (RNE) method proposed by Luh et al. (1980), which has linear complexity $O(n)$ but requires full knowledge of the inertial parameters $\mathbf{X}$, and thus is difficult to calibrate experimentally. However, modifications of the RNE algorithm, proposed by Atkeson et al. (1986), allow rearrangement into regressor form while maintaining linear complexity $O(n)$. These algorithms can be categorized into either numerical or symbolic algorithms. Numerical algorithms such as the RNE method work "out of the box" and can provide results using only the kinematic and inertial parameters of the kinematic chain, without a derivation step. Conversely, symbolic algorithms, such as the Lagrangian approach, require a complex derivation step, but can eliminate redundant terms and simplify calculations. For industrial robots, the number of links is typically small (six or seven joints), and as such, these simplifications can result in lower evaluation time than numerical algorithms *even though the algorithm complexity is higher* (Lloyd et al., 2021a). However, these symbolic programs typically struggle with the complex derivations required and lack robustness — relying on algorithms that use significant memory and can fail or give sub-optimal results. Moreover, common symbolic software such as *Maple* or *Mathematica* require expensive licenses to use, which practically limits access to their use.

⋆ This work was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC) grants RGPIN-2017-06967, RGPIN-2015-04169, and CRDPJ 514258-17. Support for Steffan Lloyd was provided by NSERC CGS-D.

10.1016/j.ifacol.2022.11.164

Recently, Lloyd et al. (2021a) proposed a new approach to dynamic modeling called *Pseudo-Symbolic Dynamic Modeling*, or PSDM. The PSDM method is a middle ground between symbolic and numerical methods. It is completely numerical, relying solely on standard linear algebra operations. However, it is able to perform simplifications and reductions typically only possible using symbolic software. Additionally, the derivation step is significantly faster and more reliable than with symbolic software, and the derived model is returned in regressor form, allowing for straightforward calibration. The resulting dynamic model is encoded numerically in matrix form, which allows symbolic-style manipulations to be performed on the model using standard linear algebra operations. Thus, the model is flexible to use and manipulate, allowing for highly optimized real-time code generation and forward dynamic modeling.

The goal of the current work is to demonstrate the utility of the PSDM method in the modeling, calibration, and real-time use of a dynamic model for a Denso VS-6556G, 6-DOF articulated robot for use in deburring applications. An overview of the PSDM method is given, and the derivation of the PSDM model is completed for the Denso manipulator. A procedure to calibrate this model is proposed and demonstrated over a series of experimental tests. The calibration also includes identifying the torque constants of the robot actuators, and a LuGre dynamic friction model for each joint. This article represents the first experimental validation of the PSDM method, quantifying the accuracy of the resulting models and comparing their real-time speed against other state-of-the-art methods. This paper is organized as follows. Section 2 summarizes the PSDM method and demonstrates its application to our deburring robot. Motor and friction models for the dynamic calibration are also introduced. Section 3 presents an experimental approach to calibrate the derived model. Finally, Section 4 demonstrates the speed benefits of PSDM over existing algorithms.

2. PSEUDO-SYMBOLIC DYNAMIC MODELING

The PSDM method (Lloyd et al., 2021a) is a new method of deriving, representing, and computing the dynamic model of a serial kinematic chain. It is not directly based on either the Lagrange or Newton-Euler formulations of dynamics. PSDM is motivated by the observation, proven to be true for all serial kinematic chains in Lloyd et al. (2021a), that the equations of motion, when expanded out fully, take on an exceedingly regular form. The torque in a given joint i can be shown to be the sum of p terms

$$\tau_i = \theta_1\, \upsilon_1(\mathbf{q})\, a_1(\dot{\mathbf{q}}, \ddot{\mathbf{q}}) + \cdots + \theta_p\, \upsilon_p(\mathbf{q})\, a_p(\dot{\mathbf{q}}, \ddot{\mathbf{q}}), \tag{4}$$

where, for each j^{th} summation function,

- θ_j is a lumped inertial parameter consisting of some combination of the original inertial parameters $\mathbf{X}$,
- $\upsilon_j(\mathbf{q})$ is a member of the set of functions known as the *geometric multiplier functions of order 2* of the manipulator, which are formed from the trigonometric relationships of each joint angle q_i. Practically, for an articulated robot, this means that $\upsilon_j(\mathbf{q})$ is the product of one function from the set $\mathcal{Z}^{(2)}_{q_i} = \{1,\ \sin(q_i),\ \cos(q_i),\ \sin(q_i)\cos(q_i),\ \cos^2(q_i)\}$, for each of the n joints in the manipulator, such that,
$$\upsilon_j(\mathbf{q}) = \prod_{i=1}^{n} \zeta(q_i), \qquad \zeta(q_i) \in \mathcal{Z}^{(2)}_{q_i}. \tag{5}$$
For prismatic joint robots, this definition is slightly modified – see the complete definition by Lloyd et al. (2021a) for full details.
- $a_j(\dot{\mathbf{q}}, \ddot{\mathbf{q}})$ is a function in the set of *acceleration functions* of the manipulator — in other words, $a_j(\dot{\mathbf{q}}, \ddot{\mathbf{q}})$ is either one of the n joint accelerations $a(\ddot{\mathbf{q}}) = \ddot{q}_i$, the $\binom{2}{n}$ Coriolis accelerations $a(\dot{\mathbf{q}}) = \dot{q}_i\dot{q}_j$, the n centrifugal accelerations $a(\dot{\mathbf{q}}) = \dot{q}_i^2$, or the single gravity acceleration, $a = g$.

This observation is leveraged in PSDM since it is possible to assemble a complete list of *all* the possible combinations of $\upsilon_j(\mathbf{q})\, a_j(\dot{\mathbf{q}}, \ddot{\mathbf{q}})$ for a given manipulator. If the corresponding inertial parameters θ_j are selected correctly, this list of functions, when summed together, could exactly represent the equations of motion of the robot. Moreover, this list could be represented in regressor form as

$$\begin{aligned}\boldsymbol{\tau}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})^{\mathsf{T}} &= [\upsilon_1 a_1\ \cdots\ \upsilon_p a_p] \begin{bmatrix} \theta_{1,1} & \cdots & \theta_{1,n} \\ \vdots & & \vdots \\ \theta_{p,1} & \cdots & \theta_{p,n} \end{bmatrix} \\ &= \mathbf{y}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})\, \boldsymbol{\Theta},\end{aligned} \tag{6}$$

where $\mathbf{y}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$ is a $1 \times p$ regressor matrix, and $\boldsymbol{\Theta}$ is an $p \times n$ matrix of various lumped inertial parameters — themselves all implicitly functions of $\mathbf{X}$.

Typically an exhaustive list of all possible combinations of functions is highly redundant. The PSDM method corrects this by reducing the function list, as in (4), in a two-step numerical process. The first step in this process is to remove all redundant functions from the list. Redundant functions can be identified because in a regression for $\boldsymbol{\Theta}$ of (6), knowing $\mathbf{y}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$ and $\boldsymbol{\tau}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$, the corresponding row of $\boldsymbol{\Theta}$ will be identically zero. Moreover, this regression can be done using synthetic, maximally exciting random samples by computing $\mathbf{y}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$ at randomly generated poses and computing the corresponding torques $\boldsymbol{\tau}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$ using any numerical algorithm such as the RNE algorithm. After this elimination step, the model will still be in the same form as (6), except that the number of functions p will be the minimum number of functions required to exactly represent the inverse dynamic model of the manipulator, and as such, the model is "simplified."

The second reduction step seeks to reduce the $p \times n$ inertial matrix $\boldsymbol{\Theta}$ into a single ℓ-vector of base inertial parameters $\boldsymbol{\theta}_b$. This reduction is performed via a secondary numerical process that identifies n pairs of *reduction matrices* $\mathbf{P}_i \in \mathbb{R}^{p \times \ell}$ and $\mathbf{B}_i \in \mathbb{R}^{\ell \times p}$, for each joint $i = 1, \ldots, n$, such that

$$\boldsymbol{\theta}_b = \mathbf{B}_i\, \boldsymbol{\theta}_i, \qquad \mathbf{P}_i \mathbf{B}_i = \mathbf{I}, \tag{7}$$

where $\boldsymbol{\theta}_i$ is the i^{th} column of $\boldsymbol{\Theta}$, such that the torque in each joint i, $i = 1, \ldots, n$, can be rewritten as

$$\begin{aligned}\tau_i &= \mathbf{y}_p(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})\, \boldsymbol{\theta}_{p,i} = \mathbf{y}_p(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})\, \mathbf{P}_i\, \mathbf{B}_i\, \boldsymbol{\theta}_{p,i} \\ &= \left[\, \mathbf{y}_p(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})\, \mathbf{P}_i \,\right] \boldsymbol{\theta}_b.\end{aligned} \tag{8}$$

Furthermore, if we form the matrix $\mathbf{Y}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}) \in \mathbb{R}^{n \times \ell}$ as

$$\mathbf{Y}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}) \triangleq \begin{bmatrix} \mathbf{y}_p(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})\, \mathbf{P}_1 \\ \vdots \\ \mathbf{y}_p(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})\, \mathbf{P}_n \end{bmatrix} \tag{9}$$

then (8) can be reformulated more compactly as

$$\boldsymbol{\tau} = \mathbf{Y}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})\, \boldsymbol{\theta}_b. \tag{10}$$

The process of deriving the reduction matrices $\mathbf{P}_i$ and $\mathbf{B}_i$ is omitted from the current work for brevity. The process is accomplished using a secondary numerical analysis, again using the RNE method to "sample" the linear relationship, then performing a QR decomposition to analyze dependencies between the inertial parameters. The reduction matrices are *not* unique and can be related by any similarity transformation by a non-singular $p \times p$ matrix. However, the method proposed by Lloyd et al. (2021a) ensures that the resulting matrices are highly sparse, thus maximizing the efficiency of the model.

Once these reduction matrices have been found, the PSDM derivation is complete. Eq. (8) is in simplified, regressor form and can be calibrated experimentally via appropriate selection of the ℓ-vector of base inertial parameters $\boldsymbol{\theta}_b$.

2.1 Derivation for the Denso VS-6556G Manipulator

In the current work, we are concerned with the dynamic modeling of a Denso VS-6556G manipulator, outfitted for robotic deburring as shown in Fig. 1, and whose kinematic parameters are given in Table 1. The implementation of the PSDM algorithm is quite complex, and indeed many additional checks and simplifications go into the derivation process that there is not space in the current paper to discuss. However, the PSDM algorithm has already been implemented in Lloyd et al. (2021a) and is readily used with only a few lines of Matlab code. Using the `PSDM.deriveModel` function in the provided codebase, the derivation steps run sequentially, and the function returns the simplified model in a few seconds. The result is two arrays that completely describe the PSDM model — a $5n \times p$ integer array `E`, which numerically encodes the list of minimal functions that make up the variable $\mathbf{y}_p(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$, and a $p \times \ell \times n$ array `P`, of the model's n reduction matrices.

For the Denso VS-6556G manipulator, we run the derivation function with the following input settings:

- Drive inertias are included in the analysis for all links, denoted as I_{mi}, $i = 1, \ldots, n$;
- Tool inertial properties are excluded since the tool can be properly weighed and calibrated separately.
- Link 6 is nearly massless, and as such, all inertial parameters on this link are assumed to be zero.
- Links 4 and 5 have lower weight and are highly symmetrical. This symmetry means that certain inertial values are known to be negligible, and r_{x5}, I_{xz4}, I_{yz4}, I_{xy5}, I_{xz5}, and I_{yz5} are assumed to be zero.

The derivation runs in approximately 3 seconds, and the PSDM algorithm identifies $p = 1318$ minimal functions in the model and $\ell = 32$ base inertial parameters $\boldsymbol{\theta}_b$. This model is too large to list here; however, it is provided within the supplementary files for this paper, available online with the PSDM Github repository, for those wishing to use or reproduce our results (Lloyd et al., 2021b).

2.2 Motor & Friction Models

The derived PSDM model above captures the gravity and inertial effects of the multi-body linkage, relating the torques in the joints to the resulting accelerations. However, two additional models are required to use this dynamic model practically. The first is a motor model to correctly relate the measured motor currents to joint torques (as the Denso manipulator does not have joint torque sensors), and the second is a model to estimate the joint friction forces from bearing, seals, and gearboxes.

Motor Model The Denso VS-6556G robot can report the set-points of the actuator currents in its joints, and these can be used to measure torque in the joints. In the current work, we assume the dynamic relationship between current and torque (i.e., the inductance) to be negligible and use the standard linear relationship

$$\boldsymbol{\tau} = \operatorname{diag}(\mathbf{K}_t)\, \mathbf{i}, \tag{11}$$

where $\mathbf{i}$ is a vector of the joint currents, and $\mathbf{K}_t$ is a vector of the effective torque constant of actuators (after any motion conversion due to gearboxes or belts).

Friction Model To accurately model the dynamics of the Denso robot, friction must be taken into account. This robot features heavy-duty seals to prevent the ingress of water and dust, and these seals create large friction forces in the joints. Additionally, these seals deflect during operation, and the forces they impart are history-dependent. We use the LuGre friction model proposed by Canudas de Wit et al. (1995) to model these complex friction forces. The LuGre model treats the friction contact akin to the bristles of a comb. For small displacements, the bristles deflect as a spring, and "bounce back" if the force is removed. However, for larger displacements, the bristles deflect, then slide. The LuGre model introduces a dynamic state z_i into the model to track the state of friction in each joint, which describes the *mean bristle deflection*. The dynamics of this state are described by the first-order differential equation

$$\dot{z}_i = \dot{q}_i - \left|\dot{q}_i\right| z_i / g_i(\dot{q}_i), \tag{12}$$

Table 1. Denso VS-6556G DH parameter list

	Joint angle	Link offset	Link length	Link twist
1	q_1	132 mm	75 mm	$-\pi/2$ rad
2	q_2	0 mm	−270 mm	0 rad
3	q_3	0 mm	−90 mm	$\pi/2$ rad
4	q_4	295 mm	0 mm	$-\pi/2$ rad
5	q_5	0 mm	0 mm	$\pi/2$ rad
6	q_6	80 mm	0 mm	0 rad

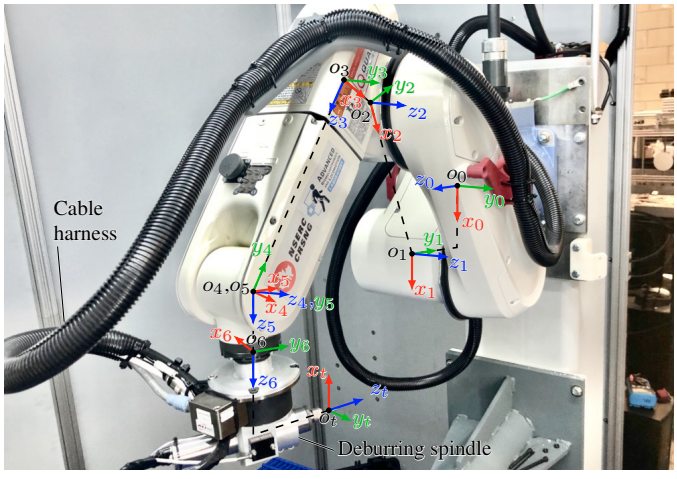

Fig. 1. The Denso VS-6556G outfitted with a spindle and force sensor for deburring applications.

where $g_i(\dot{q}_i)$ is a positive definite function describing the macro friction effects in joint i and can be tailored to the application. In the current work, the Stribeck (1903) model is used, such that

$$g_i(\dot{q}_i) = \frac{1}{\sigma_{0i}} \left[F_{ci} + \left(F_{si} - F_{ci} \right) e^{-\left| \dot{q}_i / v_{si} \right|^2} \right], \tag{13}$$

where σ_{0i} is the bristle stiffness, F_{ci} is the Coulomb (dry friction) constant, F_{si} is the static friction constant, and v_{si} is the transition velocity (where static friction forces transition to dynamic friction forces). The friction torques can then be computed as the sum of the bristle bending and damping torques, plus the viscous friction effects, as

$$\tau_{\text{fric},i} = \sigma_{0i}\, z_i + \sigma_{1i}\, \dot{z}_i + F_{vi} \left| \dot{q}_i \right|^{d_{vi}} \operatorname{sign} \dot{q}_i, \tag{14}$$

where F_{vi} is the viscous friction constant, and d_{vi} is a shape parameter that allows the viscous friction curve to be warped to better fit the observed friction over a broader range of speeds (Olsson, 1996). With this model, small displacements will behave like a spring-damper system with constants σ_{0i} and σ_{1i}, and large displacements will tend asymptotically towards $\sigma_{0i}\, g_i(\dot{q}_i) \operatorname{sign} \dot{q}_i$.

3. DYNAMIC MODEL CALIBRATION

In Section 2, models were derived to describe the inertia, gravity, motor, and friction effects of a Denso manipulator. However, these models cannot be properly leveraged without model calibration. This calibration is divided into four separate steps, which allows for specialized experiments to be designed at each identification step that maximally excite the selected calibration parameters. As such, the dynamic equation divided up as

$$\boldsymbol{\tau} = \operatorname{diag}(\mathbf{K}_t)\, \mathbf{i} = \boldsymbol{\tau}_{\text{grav}} + \boldsymbol{\tau}_{\text{fric}} + \boldsymbol{\tau}_{\text{iner}} + \boldsymbol{\tau}_{\text{ee}}, \tag{15}$$

where the end-effector torque, $\boldsymbol{\tau}_{\text{ee}}$, is the torque induced by the known end-effector mass properties (from CAD).

3.1 Motor Calibration

The first step is to calibrate the motor models, by finding the n parameters K_{ti} as in (11). To accomplish this, we run each joint through a full sweep of its joint range, at a constant velocity, in the forward and reverse directions, as well as with and without a mass of known weight attached to the end-effector. Because the effects of friction are assumed to be the same in both joint directions, we can effectively eliminate the effects of friction by averaging the observed currents in the positive and negative directions. Then, by adding a known mass to the end-effector, we can induce a known load into the joint, and by comparing to the nominal (massless) case, we remove the effects of gravity. By keeping speeds and accelerations low, we remove the inertial effects. Thus, (15) reduces to

$$\operatorname{diag}(\mathbf{K}_t)(\mathbf{i}_w - \mathbf{i}_{\text{nom}}) = \mathbf{J}(\mathbf{q})^{\mathsf{T}}\, \mathbf{f}_w \tag{16}$$

where $\mathbf{i}_w$ is the current with the added mass and $\mathbf{i}_{\text{nom}}$ without, $\mathbf{J}(\mathbf{q})$ is the manipulator Jacobian, expressed in world frame, and $\mathbf{f}_w$ is the wrench imparted by the calibration weight. This wrench can be computed for a mass m_w, center of gravity $\mathbf{r}_w$, gravity vector $\mathbf{g}$, and rotation matrix (from world to tool frame) $\mathbf{R}(\mathbf{q})$ as

$$\mathbf{f}_w = \begin{bmatrix} m_w\, \mathbf{g} \\ \mathbf{R}(\mathbf{q})\, \mathbf{r}_w \times (m_w\, \mathbf{g}) \end{bmatrix}. \tag{17}$$

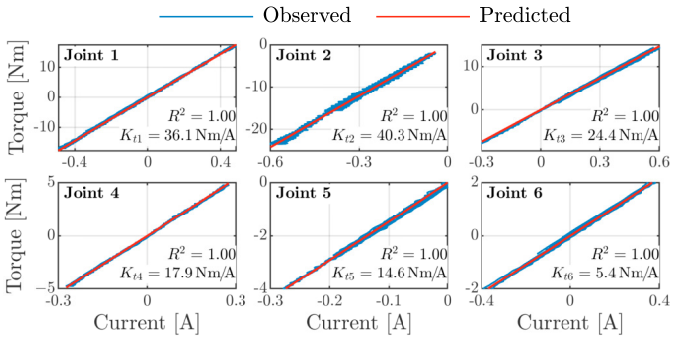

Fig. 2. The observed and predicted motor torques.

Testing was done individually in a pose designed for each joint to ensure that the calibration mass was fully observable by the target joint. Three masses were used, ranging from 1.2 to 3.5kg. For each test, the currents $\mathbf{i}$ and joint positions $\mathbf{q}$ were collected. The data was filtered non-causally with a zero-phase shift, 8$^{\text{th}}$-order Butterworth filter with a 4 Hz cutoff frequency implemented via Matlab's `filtfilt` function. The data was divided into training and validation sets with an 85/15% split before using (16) to solve for $\mathbf{K}_t$ using linear least squares. The resulting values are shown in Table 2, and the regression data is shown in Fig. 2. The accuracy of the fits is given in Table 3 and shows the calibration was successful, with the model capturing nearly 100% of the observed effects via the R^2 metric.

3.2 Gravity Calibration

The next step taken in the calibration process is identifying the gravity terms of the PSDM model. These terms are identified separately from the rest of the dynamic model since they are easier to test than the inertial parameters, are more significant to the model's accuracy, and can be tested at low velocities. Extraction of the gravity model using PSDM is straightforward since the model is given organized by the acceleration type of each summation term, so one simply needs to filter out the columns of $\mathbf{Y}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$ which use the acceleration function $a_j = g$. This reduction yields the reduced gravity regressor matrix $\mathbf{Y}_{\text{grav}}(\mathbf{q})$. This regressor will only allow observation of a small subset of $\boldsymbol{\theta}_b$, which in our case is nine parameters, denoted as $\boldsymbol{\theta}_{b,\text{grav}}$. Similar to the previous section, testing is done both forward and backward, and the resulting currents are averaged to remove the effects of friction. Testing is done at joint velocities high enough to be outside the static friction regime, but slow enough that inertial effects can be neglected. Thus, (15) reduces to

$$\begin{aligned} \boldsymbol{\tau} &= \boldsymbol{\tau}_{\text{grav}} + \boldsymbol{\tau}_{\text{ee}} \\ \operatorname{diag}(\mathbf{K}_t)\, \mathbf{i} &= \mathbf{Y}_{\text{grav}}(\mathbf{q})\, \boldsymbol{\theta}_{b,\text{grav}} + \mathbf{J}(\mathbf{q})^{\mathsf{T}}\, \mathbf{f}_{\text{ee}}, \end{aligned} \tag{18}$$

where $\mathbf{f}_{\text{ee}}$ is the wrench imparted by the end-effector due to gravity, and since the full inertial properties of the tool are known, this can be computed as in (17).

Calibration trajectories for the experiment are generated pseudo-randomly by simultaneously jogging each joint at a constant, slow velocity through its full joint range for an extended period of time, then back again along the same path. Joint 6 was removed from the analysis since, as mentioned in Section 2, the final link was assumed to be effectively massless. In this paper, we generated separate

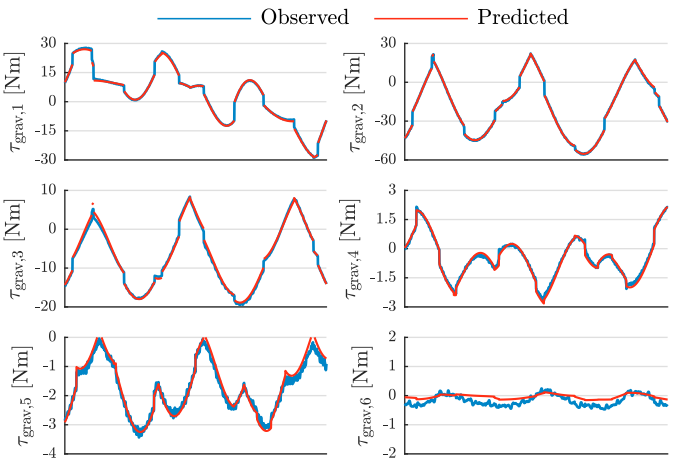

Fig. 3. Sample of observed and predicted gravity joint torques on validation set.

trajectories to identify gravity parameters of the lower "base" joints (1–3) and the upper "wrist" joints (4–5). This separation was done to mitigate collision issues during testing and prevent scaling issues during the least-squares fit. Data was collected and filtered at 4 Hz, similarly to Section 3.1, and divided into training and validation with an 85/15% split. Regression was then done using (18). The resulting identified parameters are shown in Table 4, and the accuracy of the fit on the validation sets is given in Table 3. Again, this identification step was highly accurate, capturing nearly 100% of the observed effects using the R^2 metric for the base joints and 95%/98% for the wrist joints. This accuracy can also be observed in Fig. 3, which plots the observed and predicted torques over a sample of the validation data set. The wrist joints do exhibit lower relative accuracy; however, it should be noted that the magnitude of these torques is significantly lower than the base joints, and the effect on the model is also lower. Furthermore, the nonlinear disturbing effects of the cable harness (see Fig. 1), not included in the gravity model, affect the wrist joints more than the base joints.

3.3 Friction Calibration

To test friction, we treat each joint individually and run the joint through a series of sine wave motions with varying size, ranging from small amplitudes, allowing the pre-sliding effects to be captured, to large amplitudes, capturing the effects near the speed limits of the robot. By actuating joints individually, centrifugal and Coriolis effects are zero, and so long as joint accelerations are not excessive, these can be neglected as well. Then, the torque breakdown in (15) can be arranged to isolate friction as

$$\begin{aligned}\boldsymbol{\tau}_{\text{fric}} &= \boldsymbol{\tau} - \boldsymbol{\tau}_{\text{grav}} - \boldsymbol{\tau}_{\text{ee}} \\ &= \operatorname{diag}(\mathbf{K}_t)\mathbf{i} - \mathbf{Y}_{\text{grav}}(\mathbf{q})\,\boldsymbol{\theta}_{b,\text{grav}} - \mathbf{J}(\mathbf{q})^{\mathsf{T}}\,\mathbf{f}_{\text{ee}}. \end{aligned} \quad (19)$$

In our friction model, seven parameters must be identified per joint: the Coulomb friction F_{ci}, the static friction F_{si}, the transition velocity v_{si}, the viscous constant F_{vi}, the viscous friction shape parameter d_{vi}, and finally, the LuGre pre-sliding stiffness and damping, σ_{0i} and σ_{1i}.

The robot is run through the calibration trajectories, and joint speed is computed using a central difference approximation to avoid phase shift, e.g., $\dot{q}_i[k] = \frac{q_i[k+1]-q_i[k-1]}{2T}$

Table 2. Identified motor & friction parameters.

	K_{ti} [Nm/A]	F_{ci} [Nm]	F_{si} [Nm]	v_{si} [mrad/s]	F_{vi} [Nms/rad]	d_{vi}	σ_{0i} [kNm/rad]	σ_{1i} [kNms/rad]
1	36.1	9.59	12.5	14.4	23.7	0.61	288	22.8
2	40.3	8.86	9.92	20.1	28.7	0.62	250	21.2
3	24.4	6.20	7.77	32.0	10.7	0.73	130	19.2
4	17.9	3.59	4.00	210	2.38	0.63	5.36	8.91
5	14.6	2.39	2.67	81.1	4.50	0.59	35.9	13.9
6	5.42	1.60	1.53	10.4	1.61	0.74	13.0	10.3

(where T is the sample time). Data is filtered with zero-phase shift as before, but with a higher cutoff frequency of 40 Hz. Frictional torque at each time step was then computed using (19). Data is split into training/validation with an 85/15% split. Fitting the friction terms cannot be done linearly, as with the earlier sections. Instead, the following two-step procedure is followed. At high velocities, where $\dot{q}_i \gg v_{si}$, the pre-sliding effects are negligible and can be ignored, reducing the model to

$$\tau_{\text{fric},i} \approx \left(F_{ci} + F_{vi}|\dot{q}_i|^{d_{vi}}\right) \operatorname{sign} \dot{q}_i. \quad (20)$$

Isolating the high-speed samples from the data, F_{ci}, F_{vi}, and d_{vi} can be fit for each joint. In the current work, this fitting is done using a *trust-region* nonlinear least squares method, as implemented by Matlab's `lsqnonlin` function. The pre-sliding parameters F_{si}, v_{si}, σ_{0i}, and σ_{1i} are then fit using the derivative-free Nelder-Mead simplex method (Lagarias, Jeffrey et al., 1998), implemented via Matlab's `fminsearch` method, with a cost function that simulates the LuGre dynamic equations, (12) to (14), over the entire dataset, then computes the frictional torque and returns the sum of square error over all samples. The resulting friction coefficients are given in Table 2, and the model accuracy in validation is given in Table 3. The calibration captures 98–99% of the observed friction effects by the R^2 metric. A sample of the model performance on the validation set is shown in Fig. 4. This plot shows the model accurately capturing the friction torques in high-velocity cases, where viscous effects dominate, but also in the pre-sliding regime, where hysteresis effects are large.

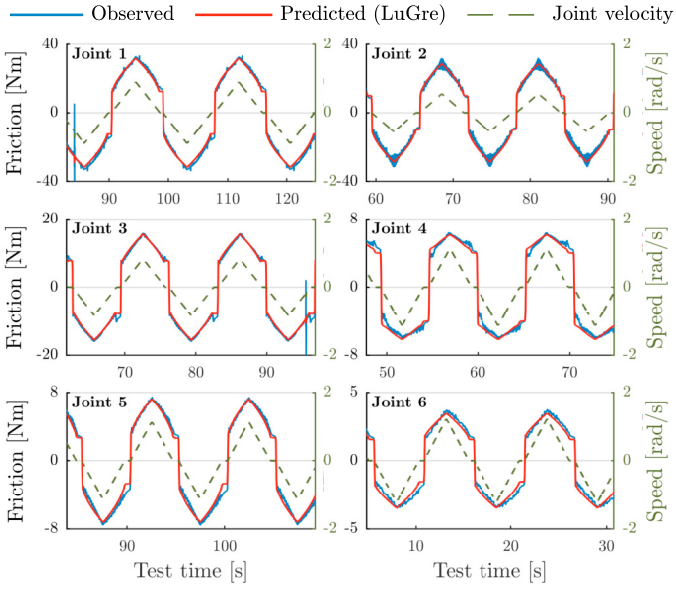

Fig. 4. Sample of observed and predicted frictional torques in validation dataset.

Table 3. Validation accuracy of calibrated motor, gravity, friction & inertial models.

	Motor		Gravity		Friction		Inertial		All‡
	RMSE	R^2	RMSE	R^2	RMSE	R^2	RMSE	R^2	R^2
1	0.24	1.00	0.30	1.00	2.16	0.99	2.16	0.78	0.99
2	0.46	1.00	0.41	1.00	3.06	0.98	2.79	0.84	0.97
3	0.35	1.00	0.55	1.00	0.61	1.00	3.02	0.35	0.86
4	0.06	1.00	0.13	0.98	0.52	0.99	0.95	0.05	0.91
5	0.12	1.00	0.18	0.95	0.49	0.99	0.55	0.40	0.96
6†	0.04	1.00	0.16	—	0.27	0.99	0.90	0.06	0.81

RMSE is the root mean squared error, reported in Nm.
† R^2 value for link 6 gravity model is not reported since the inertial properties of link 6 are assumed negligible.
‡ As evaluated on a high-acceleration, high-speed path. Slow-speed performance would be closer to the gravity performance.

3.4 Inertial Calibration

The final step to the dynamic calibration is identifying the terms of the dynamic model required to calibrate the torque components related to centrifugal, Coriolis, and joint accelerations. In this step, the inertial torques $\boldsymbol{\tau}_{\text{iner}}$ must be estimated by using the existing motor, gravity, and friction models, substituted into (15). We define $\mathbf{Y}_{\text{iner}}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$ and $\boldsymbol{\theta}_{b,\text{iner}} \in \mathbb{R}^{23}$ as the complements of $\mathbf{Y}_{\text{grav}}(\mathbf{q})$ and $\boldsymbol{\theta}_{b,\text{grav}}$, such that the remaining parameters of $\boldsymbol{\theta}_b$ not already identified during the gravity calibration can be determined as the solution to the linear equation,

$$\boldsymbol{\tau} - \boldsymbol{\tau}_{\text{grav}} - \boldsymbol{\tau}_{\text{fric}} - \boldsymbol{\tau}_{\text{ee}} = \mathbf{Y}_{\text{iner}}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})\, \boldsymbol{\theta}_{b,\text{iner}}. \tag{21}$$

In this paper, we generate trajectories for the identification of (21) with a sum of three sine waves along each joint,

$$q_i(t) = \sum_{k=1}^{3} A_{ik} \sin(\omega_{ik} t + \phi_{ik}), \tag{22}$$

where the amplitudes A_{ik}, frequencies ω_{ik}, and phases ϕ_{ik} were found through a nonlinear optimization process to maximize the condition number of the regressor matrix $\mathbf{Y}^*_{\text{iner}}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$, formed by vertically stacking $\mathbf{Y}_{\text{iner}}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$ for all samples in the trajectory, while staying within the position, speed, and acceleration limits of the joints,

$$\begin{aligned} \min_{A_{ik}, \omega_{ik}, \phi_{ik}} \quad & \text{cond}\left[\mathbf{Y}^*_{\text{iner}}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})\right], \\ \text{such that} \quad & |q_i| < q_{\text{lim},i},\ |\dot{q}_i| < \dot{q}_{\text{lim},i},\ |\ddot{q}_i| < \ddot{q}_{\text{lim},i}. \end{aligned} \tag{23}$$

This problem was optimized using a genetic algorithm, implemented via Matlab's `ga` function. A sample of the resulting amplitudes, frequencies, and phase shifts are shown in a sample of this trajectory in Fig. 5. A total of 1492 seconds of data was recorded. Speed and acceleration estimates were obtained through central difference estimation, as with the friction modeling. The data was filtered as earlier at a cutoff of 15 Hz. An 85/15% split of training and validation data was used. The model parameters were then obtained through the linear least-squares solution to (21). These are listed in Table 4, and the validation accuracy of the model is given in Table 3. Finally, a sample of the model predictions is shown in Fig. 5. We note that the accuracy of this fit is lower than the previous models. The first reason for this is a stackup of errors. In isolating the inertial torques in (21), the errors from all previous models are compounded, giving a lower accuracy in this final calibration step. The second reason is that the inertial torques are less observable than the other model effects. Even though the current testing methodology collected data while driving the actuator joints near their speed and acceleration limits, the inertial effects were still small relative to the gravity and friction effects. This difference was especially true for the wrist joints, whose torques are dominated by frictional and end-effector effects. However, this low observability also means that inertial effects have a small effect on the accuracy of the combined model, such that the full model remains accurate. It is suspected that much of the remaining error is due to the nonlinear effects of the cable harness, which are not included in the model.

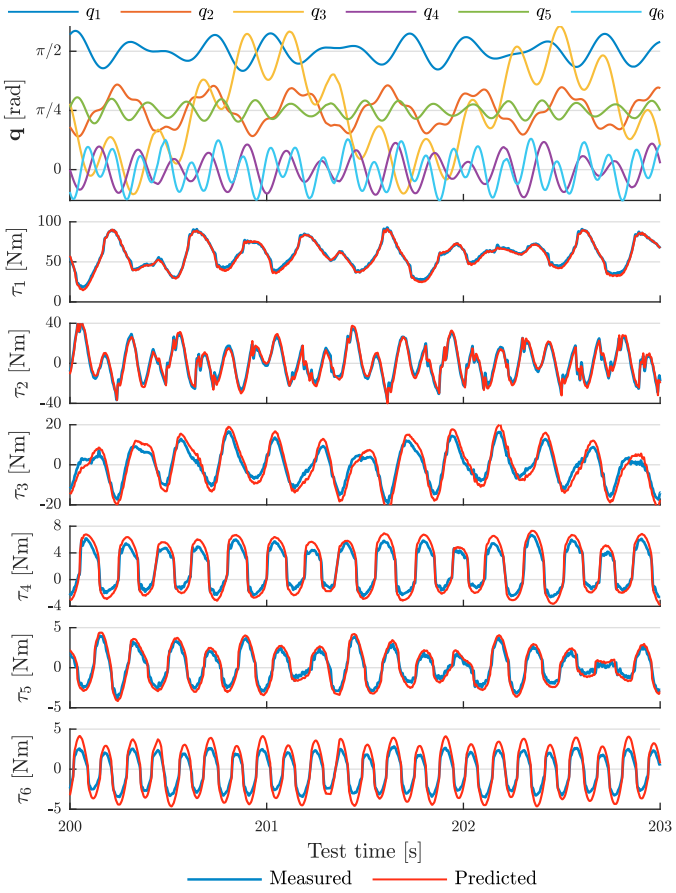

Fig. 5. *Top*: Sample of the calibration trajectory followed for acceleration testing. *Bottom*: Sample of observed and predicted torques during acceleration testing.

4. MODEL BENCHMARKING

In addition to providing a simplified model in regressor form, a significant advantage of the PSDM method is its ability to generate fast real-time code and perform forward dynamic modeling efficiently. To demonstrate this, we perform CPU benchmarks on the derived robot model. PSDM model code is provided by the real-time code generation functionality provided by the codebase in (Lloyd et al., 2021a). We provide two algorithms for comparison: the reverse Newton-Euler algorithm by Luh et al. (1980), which notably does *not* return the model in regressive form and thus is challenging to calibrate effectively, as well as the regressive form of the Newton-Euler algorithm by Atkeson et al. (1986), which is slower, but does return the model in regressive form as with the PSDM algorithm. Note as well that, unlike the PSDM algorithm, that neither of these algorithms directly allows for forward dynamic modeling — in both cases, the forward dynamic model is computed

Table 4. Identified inertial parameters θ_b.

	Estimate	Equation†
Gravity Parameters		
θ_1	7.42 kg	$m_1 + m_1 r_{x1}/a_1 - m_2 r_{x2}/a_2$
θ_2	0.566 kgm	$m_1 r_{z1} + m_2 r_{z2} + m_3 r_{y3}$
θ_4	5.46 kg	$m_2 + m_2 r_{x2}/a_2 - m_3 r_{x3}/a_3$
θ_5	−0.0632 kgm	$m_2 r_{y2}$
θ_{11}	4.1 kg	$m_3 - (m_3 r_{z3} - m_4 r_{y4})/d_4 + m_3 r_{x3}/a_3$
θ_{12}	3.38 kg	$(m_3 r_{z3} - m_4 r_{y4})/d_4 + m_4 + m_5$
θ_{19}	−0.0165 kgm	$m_4 r_{z4} + m_5 r_{y5}$
θ_{20}	−0.0179 kgm	$m_4 r_{x4}$
θ_{27}	0.0793 kgm	$m_5 r_{z5}$
Inertial Parameters		
θ_3	2.06 kgm^2	$I_{m1} + I_{yy1} + I_{xx2} + I_{xx3} + I_{xx4} + I_{zz5} + m_1({a_1}^2 + {r_{x1}}^2 + {r_{z1}}^2 + 2a_1 r_{x1}) + m_2({r_{y2}}^2 + {r_{z2}}^2 - {a_1}^2 r_{x2}/a_2) + m_3({r_{y3}}^2 + {r_{z3}}^2 - d_4 r_{z3}) + m_4({r_{y4}}^2 + {r_{z4}}^2 - d_4 r_{y4}) + m_5 {r_{y5}}^2$
θ_6	2.82 kgm^2	$2m_2 {r_{y2}}^2 + I_{m2} + I_{xx2} - I_{yy2} + I_{zz2}$
θ_7	2.66 kgm^2	$m_2 {r_{x2}}^2 + m_2 a_2 r_{x2} + m_2 {r_{y2}}^2 + I_{m2} + I_{zz2}$
θ_8	0.0177 kgm^2	$I_{yz2} - m_2 r_{y2} r_{z2}$
θ_9	0.1 kgm^2	$I_{xz2} - m_2(a_2 r_{z2} - r_{x2} r_{z2}) - m_3 a_2 r_{y3}$
θ_{10}	−0.236 kgm^2	$I_{xy2} - m_2 r_{x2} r_{y2}$
θ_{13}	−0.0952 kgm^2	$2m_4({r_{y4}}^2 - d_4 r_{y4} + {r_{z4}}^2) + 2m_3({r_{z3}}^2 - d_4 r_{z3}) + 2m_5 {r_{y5}}^2 + I_{xx3} + I_{yy3} - I_{zz3} + 2I_{xx4} + 2I_{zz5}$
θ_{14}	−0.0443 kgm^2	$m_3({r_{x3}}^2 + a_3 r_{x3} + {r_{z3}}^2 - d_4 r_{z3}) + m_5 {r_{y5}}^2 + m_4({r_{y4}}^2 - d_4 r_{y4} + {r_{z4}}^2) + I_{yy3} + I_{xx4} + I_{zz5}$
θ_{15}	0.0397 kgm^2	$I_{xz3} - m_3 r_{x3} r_{z3}$
θ_{16}	−0.0199 kgm^2	$I_{xy3} - m_3(a_3 r_{y3} - r_{x3} r_{y3})$
θ_{17}	−0.0123 kgm^2	$I_{yz3} - m_3 r_{y3} r_{z3}$
θ_{18}	0.561 kgm^2	I_{m3}
θ_{21}	−0.0573 kgm^2	$2m_4 {r_{z4}}^2 + 2m_5 {r_{y5}}^2 + I_{xx4} + I_{yy4} - I_{zz4} + 2I_{zz5}$
θ_{22}	−0.0474 kgm^2	$-m_4({r_{x4}}^2 + {r_{z4}}^2) + m_5 {r_{y5}}^2 + I_{xx4} - I_{zz4} + I_{zz5}$
θ_{23}	−0.0172 kgm^2	$I_{xy4} - m_4 r_{x4} r_{y4}$
θ_{24}	0.00447 kgm^2	$m_4 r_{y4} r_{z4}$
θ_{25}	−0.0704 kgm^2	$m_4 r_{x4} r_{z4}$
θ_{26}	0.0838 kgm^2	I_{m4}
θ_{28}	−0.00706 kgm^2	$I_{xx5} - I_{yy5} - I_{zz5}$
θ_{29}	0.00694 kgm^2	$m_5 {r_{z5}}^2 + I_{yy5}$
θ_{30}	−0.00622 kgm^2	$m_5 r_{y5} r_{z5}$
θ_{31}	0.092 kgm^2	I_{m5}
θ_{32}	0.0306 kgm^2	I_{m6}

† Equations for each regression parameter are matched from a separate symbolic analysis. This analysis is beyond the scope of the current paper, but the equations are given for reference.

by evaluating the inverse model n times with unit acceleration on each joint to build up the mass matrix, then an additional time to evaluate the centrifugal, Coriolis and gravity torques, before solving for the joint accelerations (Lynch and Park, 2017). More efficient formulations are possible; however, these comparisons are beyond the scope of the current work. All algorithms are coded in C-code.

Table 5 lists CPU benchmarks on the current PSDM model, in forward and inverse dynamics, and compares this to the regressive reverse Newton-Euler algorithm. Results are averaged over 10^6 samples on randomly generated states. In all cases, the PSDM algorithm significantly outperforms the RNE algorithms by a full order of magnitude or more. This difference is because while the RNE algorithm solves the general dynamics problem, the PSDM model has been simplified during the numerical

Table 5. Real-time evaluation speeds.

	Inverse Model	Forward Model
PSDM	0.40 µs	0.63 µs
RNE	6.00 µs	19.9 µs
RNE (Regressive Form)	17.2 µs	112 µs

Testing done on a laptop computer with an Intel i5-8259U CPU.

simplifications described in Section 2. It avoids computing redundant terms and takes advantage of model simplifications due to joint alignments and zero-value inertial terms. This increased efficiency is highly desirable in cases where the dynamic model must be evaluated very rapidly, such as in real-time control scenarios, high-fidelity simulations, model-based observers such as Kalman filters, or any combination thereof (such as model-based predictive control).

5. CONCLUSION

This paper gives a step-by-step procedure for the derivation and full calibration of a robotic manipulator using the PSDM algorithm, including actuator and frictional modeling. The presented calibration was highly accurate, capturing nearly 98–100% of the motor, gravity, and friction effects, and while the high-speed acceleration fitting was less accurate, the overall model still captured 81–99% of the observed torque effects. Benchmarking of the derived PSDM model showed it to be significantly faster than the reverse Newton-Euler algorithm, allowing for more flexibility in how the model is used.

REFERENCES

Atkeson, C.G., An, C.H., and Hollerbach, J.M. (1986). Estimation of inertial parameters of manipulator loads and links. *Int. J. Rob. Res.*, 5(3), 101–119. doi: 10.1177/027836498600500306.

Canudas de Wit, C., Olsson, H., Astrom, K., and Lischinsky, P. (1995). A new model for control of systems with friction. *IEEE Trans. Automat. Contr.*, 40(3), 419–425. doi:10.1109/9.376053.

Lagarias, Jeffrey, C., Reeds, James, A., Wright, Margaret, H., and Wright, Paul, E. (1998). Convergence properties of the nelder–mead simplex method in low dimensions. *SIAM J. Optim.*, 9(1), 112–147.

Lloyd, S., Irani, R., and Ahmadi, M. (2021a). A numeric derivation for fast regressive modeling of manipulator dynamics. *Mech. Mach. Theory*, 156, 104149. doi: 10.1016/j.mechmachtheory.2020.104149.

Lloyd, S., Irani, R., and Mojtaba, A. (2021b). Github Pseudo-Symbolic Dynamic Modeling repository. URL `https://github.com/CarletonABL/PSDM`.

Luh, J.Y.S., Walker, M.W., and Paul, R.P.C. (1980). On-line computational scheme for mechanical manipulators. *J. Dyn. Syst. Meas. Control*, 102(2), 69–76. doi: 10.1115/1.3149599.

Lynch, K. and Park, F. (2017). *Modern robotics: mechanics, planning and control*. Cambridge Univ. Press, USA.

Olsson, H. (1996). Control systems with friction. *Dep. Autom. Control. Lund Inst. Technol.*, 1045(October), 172. doi:10.1103/PhysRevE.51.6235.

Stribeck, R. (1903). Die wesentlichen Eigenschaften der Gleit- und Rollenlager.

Uicker, J.J. (1969). Dynamic behavior of spatial linkages. *J. Eng. Ind.*, (February), 251–265.