Supplementary Material to
Robust Locomotion Control of a Self-Balancing and Underactuated Bipedal Exoskeleton: Task Prioritization and Feedback Control

Ahmed Fahmy Soliman[1] and Barkan Ugurlu[1]

Abstract— **This is a supplementary material to *Robust Locomotion Control of a Self-Balancing and Underactuated Bipedal Exoskeleton: Task Prioritization and Feedback Control*. The document includes further technical details regarding the modified Newton-Euler algorithm that was used to compute floating base dynamics of the system. Furthermore, the recursive algorithm that we used to compute Jacobian matrices is provided.**

I. System Model

Co-Ex joint configuration is illustrated in Fig. 1. The same figure also describes the parameters that are used to compute kinematics and dynamics. Each leg possesses 4 DoFs (Degrees of Freedom), namely, a 2 DoF hip joint along the A/A (Adduction/Abduction) and F/E (Flexion/Extension) axes, a 1 DoF knee joint along the F/E axis, and a 1 DoF ankle joint along the DP/F (Dorsi/Plantar Flexion) axis.

II. Forward Kinematics and Jacobian Computation

The bipedal model can be considered as a branched structure. The floating coordinates was represented by considering 6 virtual DoFs to represent the base frame $F_{0,0}$ position and orientation with respect to the inertial frame F_G; see Fig. 1. We are interested in controlling the following coordinates: i) center position and orientation of the right foot, ii) center position and orientation of the left foot, iii) position of the CoM (Center of Mass) and torso orientation. In another words, we have 3 end-effectors.

In order to compute the related end effector Jacobian matrices in a computationally inexpensive way, a recursive Newton-Euler (NE) approach with forward propagation is employed. It has a complexity of $O(n+6)$. In this algorithm, angular and translational velocity vectors concerning the i^{th} frame are respectively yielded as below.

$$\boldsymbol{\omega}_i = \boldsymbol{\omega}_{i-1} + {}^0\mathbf{R}_i \mathbf{z}_g \dot{\boldsymbol{q}}_i \tag{1}$$
$$\boldsymbol{v}_i = \boldsymbol{v}_{i-1} + \boldsymbol{\omega}_i \times \mathbf{r}_i \tag{2}$$

In (1)-(2), ${}^0R_i$ is the orientation matrix of the i^{th} link with respect to frame $F_{0,0}$, $\mathbf{z}_g = [0\ 0\ 1]^\intercal$ is a generalized z-axis selection vector, r_i is the displacement vector between the i^{th} and $i-1^{th}$ links, and $\dot{q}_i$ represents the angular velocity of the i^{th} joint. Using (1)-(2), angular and translational portions of the corresponding Jacobian matrices can be obtained as below:

$$\mathbf{J}^i_\omega = [\mathbf{J}^{i-1}_\omega \ \ {}^0\mathbf{R}_i\mathbf{z}_g] \tag{3}$$
$$\mathbf{J}^i_v = [\mathbf{J}^{i-1}_v \ \mathbf{0}_{3\times1}] - s(\mathbf{r}_i)\mathbf{J}^i_\omega \tag{4}$$

In (3)-(4), $\mathbf{0}_{3\times1}$ is a 3×1 zero matrix, $s(\mathbf{r}_i)$ is the skew matrix of r_i which satisfies the cross product operation $\mathbf{r}_i\times$. The time derivatives of the Jacobian matrices are recursively computed as follows:

$$\dot{\mathbf{J}}^i_\omega = [\dot{\mathbf{J}}^{i-1}_\omega \ \ s(\boldsymbol{\omega}_i){}^0\mathbf{R}_i\mathbf{z}_g] \tag{5}$$
$$\dot{\mathbf{J}}^i_v = [\dot{\mathbf{J}}^{i-1}_v \ \mathbf{0}_{3\times1}] - s(\boldsymbol{\omega}_i)s(\mathbf{r}_i)\mathbf{J}^i_\omega - s(\mathbf{r}_i)\dot{\mathbf{J}}^i_\omega, \tag{6}$$

where $s(\omega_i)$ is the skew matrix of ω_i which satisfies the cross product $\omega_i\times$. The overall CoM Jacobian can be computed as in the following:

$$\mathbf{J}_{com} = \sum_{i=1}^{n} w_i \mathbf{J}^i_{com} \tag{7}$$

In (7), m_i is the i^{th} link mass, M is the total mass, w_i is the ratio of $w_i = m_i/M$, n is the number of links (the actuated joints) excluding the floating base. $\mathbf{J}^i_{com}$ stands for the CoM

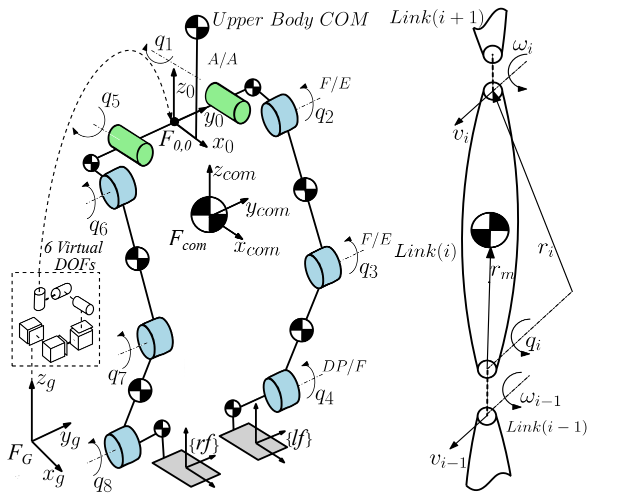

Fig. 1. Left: Joint configuration of the biped model. Right: A general manipulator structure that describes the parameters used in kinematics and dynamics algorithms.

*This work is supported by the Scientific and Technological Research Council of Turkey (TÜBİTAK), with the project 215E138.

[1]The authors are with the Biomechatronics Laboratory, Department of Mechanical Engineering, Ozyegin University, 34794 Istanbul, Turkey. fahmy.ahmed@ozu.edu.tr, barkanu@ieee.org

Jacobian concerning the i^{th} link and it can be computed recursively:

$$\mathbf{J}^i_{com} = [\mathbf{J}^{i-1}_v \ \mathbf{0}_{3\times1}] - s(\mathbf{r}^i_m)\mathbf{J}^i_\omega \tag{8}$$

Its time derivative is obtained as below:

$$\dot{\mathbf{J}}^i_{com} = [\dot{\mathbf{J}}^{i-1}_v \ \mathbf{0}_{3\times1}] - s(\boldsymbol{\omega}_i)s(\mathbf{r}^i_m)\mathbf{J}^i_\omega - s(\mathbf{r}^i_m)\dot{\mathbf{J}}^i_\omega. \tag{9}$$

In (8) and (9), $\mathbf{r}^i_m$ is the local coordinates of the i^{th} link CoM. Table I provides the algorithm to compute the Jacobian matrix and its time derivative. The subscript B indicates the floating base. The number of joints per leg is n_l; $n_l = 4$ in our case. The subscript l stands for the leg index (left: $l = 1$, right $l = 2$). The subscript b indicates the floating base virtual joints. $\mathbf{J}_{l,f}, \mathbf{J}_{r,f}$, and $\mathbf{J}_{torso}$ are the left, right foot, and torso Jacobian matrices, respectively.

III. Inverse Dynamics: The Modified Newton Euler (MNE) Algorithm

The section presents the derivation of the Modified-NE (MNE) algorithm. The symbols used in this section are defined in Table II. The MNE algorithm was developed to compute Coriolis centrifugal matrix $\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})$ recursively. Generally speaking, the conventional NE algorithm can be used to solve inverse dynamics function ID such that

$$\boldsymbol{\tau} = ID(\ddot{\mathbf{q}}, \dot{\mathbf{q}}, \mathbf{q}, \boldsymbol{\lambda}_e). \tag{10}$$

The sum of Coriolis centrifugal and gravitational terms can be obtained from (10) as below:

$$\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q}) = ID(\ddot{\mathbf{q}} = \mathbf{0}, \dot{\mathbf{q}}, \mathbf{q}, \boldsymbol{\lambda}_e = \mathbf{0}). \tag{11}$$

The eq. (11) can be further used to obtain the gravitational matrix by assigning joint velocities to zero. Thus $\mathbf{G}(\mathbf{q}) = ID(\ddot{\mathbf{q}} = \mathbf{0}, \dot{\mathbf{q}} = \mathbf{0}, \mathbf{q}, \boldsymbol{\lambda}_e = \mathbf{0})$. Subtracting $\mathbf{G}(\mathbf{q})$ from (11) yields $\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}}$. As it is required to decouple $\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})$ from the generalized joint velocity vector $\dot{\mathbf{q}}$, a set of modifications should be applied to the conventional floating base NE algorithm.

As provided in Table III, the conventional NE algorithm consists of three main loops. The first loop is a forward recursion loop covered between the 3rd and 10th lines. It is utilized to compute the CoM velocity, acceleration for each link. It additionally provides the forces that are required to produce such accelerations. The second loop is a backward recursion loop covered between the 11th and 19th lines. It is utilized to determine the forces transmitted across the joints. It also yields the inertia matrices thar are projected from children links to parent links. The last loop is a forward recursion loop covered between the 21st and 26th lines. It is used to obtain the projected floating base acceleration for each link. In line 27, the ID solution is completed by computing the torque $\boldsymbol{\tau}_i$ that is associated with the joint i. See [1] further details.

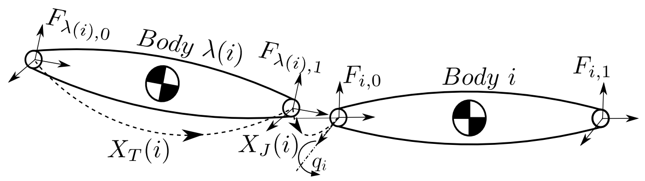

Fig. 2. Coordinate frames and transformations concerning the i^{th} joint.

```
Jacobian Computation Algorithm
1   Initially set $\mathbf{J}^0_{\omega,b}$, $\mathbf{J}^0_{v,b}$, $\dot{\mathbf{J}}^0_{\omega,b}$, and $\dot{\mathbf{J}}^0_{v,b}$ as empty matrices
2   for $i = 1$ to 6
3     if $i \leq 3$
4       $\mathbf{J}^i_{\omega,b} = [\mathbf{J}^{i-1}_{\omega,b} \ \ \mathbf{0}_{3\times1}]$
5       $\mathbf{J}^i_{v,b} = [\mathbf{J}^{i-1}_{v,b} \ {}^0\mathbf{R}_i\mathbf{z}_g] - s(\mathbf{r}_i)\mathbf{J}^i_{\omega,b}$
6       $\dot{\mathbf{J}}^i_{\omega,b} = [\dot{\mathbf{J}}^{i-1}_{\omega,b} \ \ \mathbf{0}_{3\times1}]$
7       $\dot{\mathbf{J}}^i_{v,b} = [\dot{\mathbf{J}}^{i-1}_{v,b} \ s(\boldsymbol{\omega}_i)^0\mathbf{R}_i z_g] - s(\boldsymbol{\omega}_i)s(\mathbf{r}_i)\mathbf{J}^i_{\omega,b} - s(\mathbf{r}_i)\dot{\mathbf{J}}^i_{\omega,b}$
8     else
9       $\mathbf{J}^i_{\omega,b} = [\mathbf{J}^{i-1}_{\omega,b} \ \ {}^0\mathbf{R}_i\mathbf{z}_g]$
10      $\mathbf{J}^i_{v,b} = [\mathbf{J}^{i-1}_{v,b} \ \mathbf{0}_{3\times1}] - s(\mathbf{r}_i)J^i_{\omega,b}$
11      $\dot{\mathbf{J}}^i_{\omega,b} = [\mathbf{J}^{i-1}_{\omega,b} \ \ s(\boldsymbol{\omega}_i)^0\mathbf{R}_i z_g]$
12      $\dot{\mathbf{J}}^i_{v,b} = [\dot{\mathbf{J}}^{i-1}_{v,b} \ \mathbf{0}_{3\times1}] - s(\boldsymbol{\omega}_i)s(\mathbf{r}_i)\mathbf{J}^i_{\omega,b} - s(\mathbf{r}_i)\dot{\mathbf{J}}^i_{\omega,b}$
13    end
14  end
15  $\mathbf{J}_{v,B} = \left[\mathbf{0}_{3\times2n_l} \ \mathbf{J}_{v,b}\right]$, $\mathbf{J}_{\omega,B} = \left[\mathbf{0}_{3\times2n_l} \ \mathbf{J}_{\omega,b}\right]$
16  $\dot{\mathbf{J}}_{v,B} = \left[\mathbf{0}_{3\times2n_l} \ \dot{\mathbf{J}}_{v,b}\right]$, $\dot{\mathbf{J}}_{\omega,B} = \left[\mathbf{0}_{3\times2n_l} \ \dot{\mathbf{J}}_{\omega,b}\right]$
17  for $l = 1$ to 2
18  $\mathbf{J}^0_{\omega,l} = \mathbf{J}_{\omega,b}, \ \ \mathbf{J}^0_{v,l} = \mathbf{J}_{v,b}$, and $\mathbf{J}_{com,l} = \mathbf{0}_{3\times1}$
19  $\dot{\mathbf{J}}^0_{\omega,l} = \dot{\mathbf{J}}_{\omega,b}, \ \ \dot{\mathbf{J}}^0_{v,l} = \dot{\mathbf{J}}_{v,b}$, and $\dot{\mathbf{J}}_{com,l} = \mathbf{0}_{3\times1}$
20    for $i = 1$ to $n_l$
21      $\mathbf{J}^i_{\omega,l} = [\mathbf{J}^{i-1}_{\omega,l} \ \ {}^0\mathbf{R}_{i,l}\mathbf{z}_g]$
22      $\dot{\mathbf{J}}^i_{\omega,l} = [\dot{\mathbf{J}}^{i-1}_{\omega,l} \ \ s(\boldsymbol{\omega}_i)^0\mathbf{R}_{i,l}\mathbf{z}_g]$
23      $\mathbf{J}^i_{v,l} = [\mathbf{J}^{i-1}_{v,l} \ \mathbf{0}_{3\times1}] - s(\mathbf{r}_i)\mathbf{J}^i_{\omega,l}$
24      $\dot{\mathbf{J}}^i_{v,l} = [\dot{\mathbf{J}}^{i-1}_{v,l} \ \mathbf{0}_{3\times1}] - s(\boldsymbol{\omega}_i)s(\mathbf{r}_i)\mathbf{J}^i_{\omega,l} - s(\mathbf{r}_i)\dot{\mathbf{J}}^i_{\omega,l}$
25      $\mathbf{J}^i_{com,l} = [\mathbf{J}^{i-1}_{v,l} \ \mathbf{0}_{3\times1}] - s(\mathbf{r}^i_m)\mathbf{J}^i_{\omega,l}$
26      $\dot{\mathbf{J}}^i_{com,l} = [\dot{\mathbf{J}}^{i-1}_{v,l} \ \mathbf{0}_{3\times1}] - s(\boldsymbol{\omega}_i)s(\mathbf{r}^i_m)\mathbf{J}^i_{\omega,l}$
                 $-s(\mathbf{r}^i_m)\dot{\mathbf{J}}^i_{\omega,l}$
27      $\mathbf{J}_{com,l} = m_i\mathbf{J}^i_{com,l} + \mathbf{J}_{com,l}$
28      $\dot{\mathbf{J}}_{com,l} = m_i\dot{\mathbf{J}}^i_{com,l} + \dot{\mathbf{J}}_{com,l}$
29    end
30  end
31  $\mathbf{J}_{com} = (m_b\mathbf{J}_{v,B} + \mathbf{J}_{com,1} + \mathbf{J}_{com,2})/M$
32  $\dot{\mathbf{J}}_{com} = (m_b\dot{\mathbf{J}}_{v,B} + \dot{\mathbf{J}}_{com,1} + \dot{\mathbf{J}}_{com,2})/M$
33  $\mathbf{J}_{l,f} = \begin{bmatrix}\mathbf{J}_{v,1}\\ \mathbf{J}_{\omega,1}\end{bmatrix}$, $\mathbf{J}_{r,f} = \begin{bmatrix}\mathbf{J}_{v,2}\\ \mathbf{J}_{\omega,2}\end{bmatrix}$, $\mathbf{J}_{torso} = \mathbf{J}_{\omega,B}$
34  $\dot{\mathbf{J}}_{l,f} = \begin{bmatrix}\dot{\mathbf{J}}_{v,1}\\ \dot{\mathbf{J}}_{\omega,1}\end{bmatrix}$, $\dot{\mathbf{J}}_{r,f} = \begin{bmatrix}\dot{\mathbf{J}}_{v,2}\\ \dot{\mathbf{J}}_{\omega,2}\end{bmatrix}$, $\dot{\mathbf{J}}_{torso} = \dot{\mathbf{J}}_{\omega,B}$
```

TABLE I: Jacobian computation algorithm.

The NE algorithm displayed in Table III was modified to obtain $\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})$ and $\mathbf{G}(\mathbf{q})$ separately and we call it the MNE (Modified NE) algorithm. It is provided in Table IV. We modified line 2 in the NE algorithm by adding two matrices: $\mathbf{J}_0$ and $\dot{\mathbf{J}}_0$. Initially, $\mathbf{J}_0$ is defined as an identity matrix ($\mathbf{1}_{6\times6}$), and $\dot{\mathbf{J}}_0$ is defined as a zero matrix

Symbol	Definition
$\boldsymbol{\tau}$	Joint torque vector
$(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$	Joint position, velocity, and acceleration vectors
$\boldsymbol{\lambda}_{e,i}$	Spatial force vector specifying the external force acting on body i, expressed in $F_{i,0}$.
$\boldsymbol{\lambda}_{e}$	A vector that stores the external forces acting on the bodies.
$_0$	Subscript zero indicates the floating base
$\mathbf{g}$	Gravitational acceleration
i	Actuated joint/body index. The joint i is the joint that connects body i to its parent; see Fig. 2
$\lambda(i)$	The body number of the parent of body i
N	An integer specifying the number of bodies in the mechanism
$\mathbf{X}_J(i)$	The coordinate transform from frame $F_{\lambda(i),1}$ to frame $F_{i,0}$ as shown in Fig. 2
$\mathbf{X}_T(i)$	The Plücker coordinate transform from frame $F_{\lambda(i),0}$ to frame $F_{\lambda(i),1}$.
${}^{i}\mathbf{X}_{\lambda(i)}$	The Plücker coordinate transform from frame $F_{\lambda(i),0}$ to frame $F_{i,0}$ for motion vectors
${}^{i}\mathbf{X}^{*}_{\lambda(i)}$	The Plücker coordinate transform from frame $F_{\lambda(i),0}$ to frame $F_{i,0}$ for force vectors
$\mathbf{s}_i$	Joint axis vector
h_i	Joint type (Prismatic or Revolute)
$jcalc(\bullet)$	A function used to compute joint i transformation matrix $\mathbf{X}_J(i)$ and vector s_i
$(\boldsymbol{\nu}_i, \mathbf{a}_i)$	Body i frame $F_{i,0}$ velocity and acceleration respectively
$(q_i, \dot{q}_i, \ddot{q}_i)$	Joint i position, velocity, and acceleration respectively
$\mathbf{f}_i$	The force transmitted across joint i
$\mathbf{I}_i$	Spatial inertia matrix of body i projected on $F_{i,0}$
$\times$	Cross product in motion space
$\times^*$	Cross product in force space
$\mathbf{f}_{fb}$	The force exerted by floating base on its children
$\mathbf{I}_{fb}$	Floating base Spatial inertia matrix projected on $F_{0,0}$; see Fig. 1
$(\boldsymbol{\nu}_{fb}, \mathbf{a}_{fb})$	Floating base spatial velocity and acceleration expressed in frame $F_{0,0}$
$\boldsymbol{\tau}_i$	Corresponding torque at the joint i
$\mathbf{J}_i$	Jacobian matrix for body i frame $F_{i,0}$
$\mathbf{J}_{com}$	Jacobian matrix of the aggregated CoM at frame F_{com}

TABLE II: Nomenclature for NE and MNE algorithms [1].

$(\mathbf{0}_{6\times6})$, respectively. Following this step, the linear mapping between the Coriolis & centrifugal force and velocity vector is obtained as $\boldsymbol{\tau}_{cr} = \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})[\boldsymbol{\nu}_{fb}^{\intercal} \quad \dot{\mathbf{q}}_r^{\intercal}]^{\intercal}$. The floating base velocity is denoted by $\boldsymbol{\nu}_{fb} = \mathbf{J}_{fb}\dot{\mathbf{q}}_{fb}$, where $\dot{\mathbf{q}}_{fb}$ is 6 DOFs floating base velocities. The actuated joint velocity vector is $\dot{\mathbf{q}}_r$. The joint accelerations $\ddot{q}_i$ in line 8 and the external force $\boldsymbol{\lambda}_{e,i}$ in line 9 in NE are assigned as zero vectors in MNE as stated by (11).

The lines 7 and 8 in from the NE algorithm are used to extract Jacobian, its time derivative, and gravitational acceleration for each link in the MNE algorithm; see the lines 8 and 9. The extracted matrices are then combined in line 10 to show the linear mapping matrix $\mathbf{M}_{1,i}$ defined between the force (see line 9 in the NE algorithm) and joint velocities. The line 20 and 21 in the MNE algorithm are used to transmit $\mathbf{M}_{1,i}$ and gravitational forces across the joints respectively. The line 23 in MNE is used to stack the mapping matrix $\mathbf{M}_{2,i}$ and gravitational force $\mathbf{f}_{g,i}$ for each joint into joint space matrices $\mathbf{C}_{\mathbf{r}}(\mathbf{q}, \dot{\mathbf{q}})$ and $\mathbf{G}_{\mathbf{r}}(\mathbf{q})$. The line 25 and 26 are used to augment the joint space and the floating base matrices into Coriolis & centrifugal, and gravitational matrices, namely, $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$, and $\mathbf{G}(\mathbf{q})$.

References

[1] R. Featherstone, *Rigid Body Dynamics Algorithms*. Springer US, 2008.

```
NE Algorithm
1   $\boldsymbol{\nu}_0 = \boldsymbol{\nu}_{fb}$
2   $\mathbf{a}_0 = -\mathbf{g}$
3   $for\ i = 1\ to\ N\ do$
4       $[\mathbf{X}_J, \mathbf{s}_i] = jcalc(h_i, q_i)$
5       ${}^i\mathbf{X}_{\lambda(i)} = \mathbf{X}_J \mathbf{X}_T(i)$
6       ${}^{\lambda(i)}\mathbf{X}_i^* = {}^i\mathbf{X}_{\lambda(i)}^T$
7       $\boldsymbol{\nu}_i = {}^i\mathbf{X}_{\lambda(i)}\boldsymbol{\nu}_{\lambda(i)} + \mathbf{s}_i\dot{q}_i$
8       $\mathbf{a}_i = {}^i\mathbf{X}_{\lambda(i)}\mathbf{a}_{\lambda(i)} + \mathbf{s}_i\ddot{q}_i + \boldsymbol{\nu}_i \times \mathbf{s}_i\dot{q}_i$
9       $\mathbf{f}_i = \mathbf{I}_i\mathbf{a}_i + \boldsymbol{\nu}_i \times^* \ \mathbf{I}_i\boldsymbol{\nu}_i - \boldsymbol{\lambda}_{e,i}$
10  $end$
11  $\mathbf{f}_{fb} = \mathbf{I}_{fb}\mathbf{a}_{fb} + \boldsymbol{\nu}_{fb} \times^* \ \mathbf{I}_{fb}\boldsymbol{\nu}_{fb}$
12  $for\ i = N\ to\ 1\ do$
13      $If\ \lambda(i) = 0\ then$
14      $\mathbf{I}_{fb} = \mathbf{I}_{fb} + {}^{\lambda(i)}\mathbf{X}_i^T \mathbf{I}_i \mathbf{X}_i^{\lambda(i)}$
15      $\mathbf{f}_{fb} = \mathbf{f}_{fb} + {}^{\lambda(i)}\mathbf{X}_i^* \mathbf{f}_i$
16      $else$
17      $\mathbf{I}_{\lambda(i)} = \mathbf{I}_{\lambda(i)} + {}^{\lambda(i)}\mathbf{X}_i^T \mathbf{I}_i \mathbf{X}_i^{\lambda(i)}$
18      $\mathbf{f}_{\lambda(i)} = \mathbf{f}_{\lambda(i)} + {}^{\lambda(i)}\mathbf{X}_i^* \mathbf{f}_i$
19  $end$
20  $\mathbf{a}_0 = \mathbf{a}_{fb}$
21  $for\ i = 1\ to\ N\ do$
22      $If\ \lambda(i) = 0\ then$
23          $\mathbf{a}_i = {}^i\mathbf{X}_{\lambda(i)}\mathbf{a}_{fb}$
24      $else$
25          $\mathbf{a}_i = {}^i\mathbf{X}_{\lambda(i)}\mathbf{a}_{\lambda(i)}$
26      $end$
27      $\boldsymbol{\tau}_i = \mathbf{s}_i^T(\mathbf{I}_i\mathbf{a}_i + \mathbf{f}_i)$
28  $end$
```

TABLE III: The conventional Newton Euler algorithm that can be used to compute inverse dynamics of a floating base robot [1].

```
Modified-NE (MNE) Algorithm
1   $\boldsymbol{\nu}_0 = \boldsymbol{\nu}_{fb}$
2   $\mathbf{g}_0 = -\mathbf{g},\ \mathbf{J}_0 = \mathbf{1}_{6\times 6},\ \dot{\mathbf{J}}_0 = \mathbf{0}_{6\times 6}$
3   $for\ i = 1\ to\ N\ do$
4       $[\mathbf{X}_J, \mathbf{s}_i] = jcalc(h_i, q_i)$
5       ${}^i\mathbf{X}_{\lambda(i)} = \mathbf{X}_J \mathbf{X}_T(i)$
6       ${}^{\lambda(i)}\mathbf{X}_i^* = {}^i\mathbf{X}_{\lambda(i)}^T$
7       $\boldsymbol{\nu}_i = {}^i\mathbf{X}_{\lambda(i)}\boldsymbol{\nu}_{\lambda(i)} + \mathbf{s}_i\dot{q}_i$
8       $\mathbf{J}_i = [{}^i\mathbf{X}_{\lambda(i)}\mathbf{J}_{\lambda(i)}\ \ \mathbf{s}_i]$
9       $\dot{\mathbf{J}}_i = [{}^i\mathbf{X}_{\lambda(i)}\dot{\mathbf{J}}_{\lambda}(i)\ \ \boldsymbol{\nu}_i \times \mathbf{s}_i],\ \mathbf{g}_i = {}^i\mathbf{X}_{\lambda(i)}\mathbf{g}_{\lambda(i)}$
10      $\mathbf{M}_{1,i} = (\boldsymbol{\nu}_i \times^* \ \mathbf{I}_i\mathbf{J}_i) + [\mathbf{0}_{6\times 6}\ \ \mathbf{I}_i\dot{\mathbf{J}}_i],\ \mathbf{f}_{g,i} = \mathbf{I}_i\mathbf{g}_i$
11  $end$
12  $\mathbf{f}_{fb,g} = \mathbf{I}_{fb}\mathbf{g}_o$
13  $for\ i = N\ to\ 1\ do$
14      $If\ \lambda(i) = 0\ then$
15      $\mathbf{I}_{fb} = \mathbf{I}_{fb} + {}^{\lambda(i)}\mathbf{X}_i^T \mathbf{I}_i {}^{\lambda(i)}\mathbf{X}_i$
16      $\mathbf{k}_{fb} = [\boldsymbol{\nu}_{fb} \times^* \mathbf{I}_{fb}\mathbf{J}_{\lambda(i)}\ \ \mathbf{0}_{6\times N}] + {}^{\lambda(i)}\mathbf{X}_i^* \mathbf{M}_{2,i}$
17      $\mathbf{f}_{fb,g} = \mathbf{f}_{fb,g} + {}^{\lambda(i)}\mathbf{X}^* \mathbf{f}_{g,i}$
18      $else$
19      $\mathbf{I}_{\lambda(i)} = \mathbf{I}_{\lambda(i)} + {}^{\lambda(i)}\mathbf{X}_i^T \mathbf{I}_i {}^{\lambda(i)}\mathbf{X}_i$
20      $\mathbf{M}_{2,\lambda(i)} = \mathbf{M}_{1,\lambda(i)} + {}^{\lambda(i)}\mathbf{X}_i^* \mathbf{M}_{2,i}$
21      $\mathbf{f}_{g,\lambda(i)} = \mathbf{f}_{g,\lambda(i)} + {}^{\lambda(i)}\mathbf{X}_i^* \mathbf{f}_{g,i}$
22  $end$
23  $\mathbf{C}_{\mathbf{r,i}}(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{s}_i^T \mathbf{M}_{2,i},\ \mathbf{G}_{\mathbf{r,i}}(\mathbf{q}) = \mathbf{s}_i^T \mathbf{f}_{g,i}$
24  $end$
25  $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) = [\mathbf{k}_{fb}^T\ \mathbf{C}_{\mathbf{r}}(\mathbf{q}, \dot{\mathbf{q}})^T]^T$
26  $\mathbf{G}(\mathbf{q}) = [\mathbf{f}_{fb,g}^T\ \mathbf{G}_{\mathbf{r}}(\mathbf{q})^T]^T$
```

TABLE IV: The MNE algorithm that is employed to compute Coriolis & centrifugal matrix $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ and the gravitation force vector $\mathbf{G}(\mathbf{q})$.