Multibody System Dynamics
https://doi.org/10.1007/s11044-024-10010-8

RESEARCH

The implicit inversion method for calculating the forward dynamics input Jacobian

Gabriel Krög[1] · Hubert Gattringer[1] · Andreas Müller[1]

Received: 19 January 2024 / Accepted: 24 June 2024

Abstract
This paper presents the implicit inversion method (IIM), a recursive method to evaluate the Jacobian of the forward dynamics w.r.t. the system inputs, using intermediate results obtained from an O(n) forward dynamics algorithm. The resulting coefficient matrix, called the inertia-weighted input matrix (IWIM), can be used to significantly improve the performance of solving optimal control problems that take into account system dynamics for only the current time step. As the relationship between inputs and accelerations appears fixed within a time step, this matrix can be evaluated in the initialization step of the optimization. This means that the forward dynamics only needs to be solved once at the initialization of the optimization, rather than having to solve the equations in every iteration of the optimization. The method presented in this paper especially targets a case where the forward dynamics are calculated using an O(n) method and takes advantage of variables that are already known through the evaluation of that method. These quantities allow us to obtain the inertia-weighted input matrix without having to convert the system to its generalized coordinate form. Exploiting the shape of the resulting equation, it is even possible to avoid an explicit inversion of any matrices in the process. Finally, runtime comparisons between three different methods to calculate the IWIM are made for several examples.

Keywords Dynamics · Recursive dynamics algorithms · Robotics · Numerical methods · Optimal control

1 Introduction

Task space trajectory tracking appears in different contexts [1–4], ranging from fixed-base industrial robots to legged robots or even aerial robots. The goal is to follow a prescribed task velocity $\mathbf{V}_{t,c}$ respectively acceleration $\dot{\mathbf{V}}_{t,c}$. For redundant systems, such as legged

✉ G. Krög
gabriel.kroeg@jku.at

H. Gattringer
hubert.gattringer@jku.at

A. Müller
a.mueller@jku.at

[1] Institute of Robotics, Johannes Kepler University Linz, Altenberger Straße 69, Linz, 4040, Austria

Published online: 22 July 2024

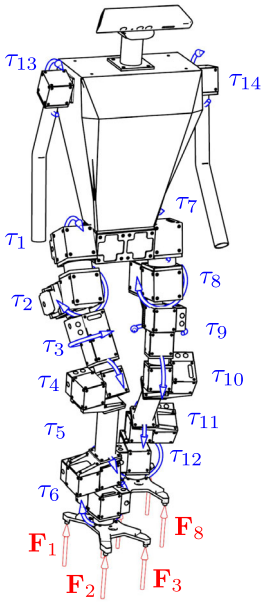

Fig. 1 Sketch of a bipedal robot showing the specified inputs. The motor torques $\tau_1, \ldots, \tau_{14}$ are shown in blue, the contact forces $\mathbf{F}_1, \ldots, \mathbf{F}_8$ between feet and ground in red (Color figure online)

robotics [5–8], the trajectory tracking problem is often formulated as an optimization problem of the form

$$\min_{\ddot{\mathbf{q}},\mathbf{u}} \frac{1}{2} \left\| \mathbf{J}_t \ddot{\mathbf{q}} + \dot{\mathbf{J}}_t \dot{\mathbf{q}} - \dot{\mathbf{V}}_{t,c} \right\|^2 \tag{1a}$$

$$\text{s.t. } \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{h}(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{B}(\mathbf{q})\mathbf{u}, \tag{1b}$$

which has to be solved for the generalized acceleration $\ddot{\mathbf{q}}$ and input $\mathbf{u}$ at the current state $(\mathbf{q}, \dot{\mathbf{q}})$. Here, $\mathbf{J}_t$ is the task Jacobian relating task velocity and generalized velocity as $\mathbf{V}_t = \mathbf{J}_t \dot{\mathbf{q}}$, and (1b) are the equations of motion (EoM) of the system with generalized mass matrix $\mathbf{M}$, input matrix $\mathbf{B}$ and $\mathbf{h}$, the sum of all remaining generalized forces. A solution of (1a)–(1b) minimizes the deviation from the desired acceleration $\dot{\mathbf{V}}_{t,c}$ while satisfying the dynamic constraints (1b). The above formulation accounts for general task tracking problem with possibly changing inputs and outputs. The inputs $\mathbf{u}$ may consist of actuation torques/forces $\boldsymbol{\tau}$ as well as interaction torques/forces $\mathbf{F}_c$ so that generally $\mathbf{u} = \begin{pmatrix} \boldsymbol{\tau}^\top & \mathbf{F}_c^\top \end{pmatrix}^\top$. As an example consider the bipedal humanoid robot in Fig. 1. The task trajectories given through $\mathbf{J}_t$ would be the trajectories of the two feet and the trajectory of its centroidal momentum, which is used to guarantee the stability of the robot [9]. Clearly, for this example, the number of prescribed task velocities and contact forces changes during the motion.

It should be emphasized that the optimization problem must be solved at a given instant of time with given state $(\mathbf{q}, \dot{\mathbf{q}})$. Moreover, of interest are the input variables $\mathbf{u}$ only rather than the acceleration $\ddot{\mathbf{q}}$. The crucial point is that a numerical optimization scheme repeatedly evaluates (1a)–(1b) for fixed state. That is, all matrices in (1b) and (1a) are constant during the optimization.

The above problem (1a)–(1b) can be transformed to an unconstrained optimization problem [9, 10]. To this end, (1b) is solved for the accelerations, also known as the forward

dynamics (FD) solution,

$$\ddot{\mathbf{q}} = -\mathbf{M}^{-1}\mathbf{h} + \mathbf{M}^{-1}\mathbf{B}\mathbf{u} = \ddot{\mathbf{q}}_0 + \ddot{\mathbf{q}}_u\,, \tag{2}$$

where $\ddot{\mathbf{q}}_0 = -\mathbf{M}^{-1}\mathbf{h}$ is the acceleration that depends on the current system state only, and $\ddot{\mathbf{q}}_u$ are the accelerations due to the inputs, $\ddot{\mathbf{q}}_u = \hat{\mathbf{B}}\mathbf{u}$. The task acceleration is thus

$$\dot{\mathbf{V}}_t = \mathbf{J}_t\ddot{\mathbf{q}} + \dot{\mathbf{J}}_t\dot{\mathbf{q}} = \mathbf{J}_t(\ddot{\mathbf{q}}_u + \ddot{\mathbf{q}}_0) + \dot{\mathbf{J}}_t\dot{\mathbf{q}} = \mathbf{J}_t\hat{\mathbf{B}}\mathbf{u} + \dot{\mathbf{V}}_{t,0}, \tag{3}$$

where $\dot{\mathbf{V}}_{t,0} = \mathbf{J}_t\ddot{\mathbf{q}}_0 + \dot{\mathbf{J}}_t\dot{\mathbf{q}}$ are the task accelerations occurring inherently from the system dynamics, which can therefore be assumed constant for the optimization since they are purely state dependent. This simplifies problem (1a) to the unconstrained optimization

$$\min_{\mathbf{u}} \frac{1}{2}\left\|\mathbf{J}_t\hat{\mathbf{B}}\mathbf{u} + \dot{\mathbf{V}}_{t,0} - \dot{\mathbf{V}}_{t,c}\right\|^2. \tag{4}$$

For solving problem (4), the EoM only need to be evaluated once for every optimization run (in which $(\mathbf{q}, \dot{\mathbf{q}})$ is const.) since except for the optimization variables $\mathbf{u}$, all other occurring quantities only depend on known values. This means that the FD only needs to be solved once to initialize the optimization. Additionally, the set of optimization variables gets smaller since the accelerations $\ddot{\mathbf{q}}$ are eliminated.

To efficiently evaluate the FD of multibody systems with many degrees of freedom (DOF), $O(n)$ methods were developed, such as the articulated-body algorithm (ABA) [11] or earlier algorithms in [12, 13], as well as a similar method presented in [14], which is used in this paper. All these methods are, however, unable to decouple the accelerations from the inputs, which is necessary for solving (4). Consequently, the IWIM $\hat{\mathbf{B}}$ should be derived separately. Explicitly evaluating and inverting the generalized mass matrix of the system to compute the IWIM as $\hat{\mathbf{B}} = \mathbf{M}^{-1}\mathbf{B}$ is computationally expensive. The performance could be improved by directly calculating the inverse of the mass matrix with a recursive method [15]. Another way to calculate the IWIM might be found through the inverse operational space inertia matrix (OSIM) $\mathbf{\Lambda}^{-1} = \mathbf{J}_t\mathbf{M}^{-1}\mathbf{J}_t^\top$, for which there are recursive algorithms [16–20]. By partitioning the input $\mathbf{u}$ into motor torques $\boldsymbol{\tau}$ and contact forces $\mathbf{F}_c$, the relationship of $\mathbf{\Lambda}^{-1}$ and $\hat{\mathbf{B}}$ can be established. Partitioning the input matrix $\mathbf{B}$ into $\mathbf{B} = \begin{pmatrix}\mathbf{J}_a^\top & \mathbf{J}_t^\top\end{pmatrix}$, with the actuation Jacobian $\mathbf{J}_a$ and the task Jacobian $\mathbf{J}_t$, yields $\mathbf{J}_t\hat{\mathbf{B}} = \begin{bmatrix}\mathbf{J}_t\mathbf{M}^{-1}\mathbf{J}_a^\top & \mathbf{\Lambda}^{-1}\end{bmatrix}$. That is, the IWIM premultiplied by the task Jacobian results in a matrix that contains the inverse OSIM. Thus, evaluating the OSIM does not help to compute the IWIM.

This paper presents the *implicit inversion method* (IIM), an algorithm which enables the calculation of the IWIM for systems with chain or tree topology using intermediate results obtained by solving an $O(n)$ method. Section 2 covers how the EoM are derived as well as the $O(n)$ algorithm that is used to solve the FD problem, showing the deficiencies inherent to this method regarding the desired application. Then, Sect. 3 presents the extended unit force method, a method to calculate the IWIM, which augments an algorithm that is originally used to evaluate the inverse OSIM. In Sect. 4 the main contribution of this work, the IIM, is presented. Building on this, simulation results of several models and runtime comparisons are shown in Sect. 5. Finally, Sect. 6 concludes this paper.

2 Subsystem formulation

To efficiently evaluate the FD of more complex systems, recursive algorithms were developed, ideally with a computational complexity of $O(n)$. In the following, an $O(n)$-method

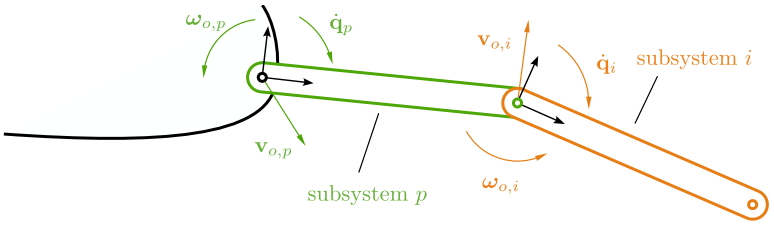

Fig. 2 Visualization of the subsystem velocity components. Both subsystems p and i are instances of the same subsystem, albeit with different parameters

[14], which is based on the subsystem formulation introduced in [14, 21], and [22], is presented. This method is equivalent to the ABA described in [11] if subsystems only consist of single bodies.

The subsystem formulation as introduced in [14, 21] is a modeling approach where the overall system is considered an assembly of smaller *subsystems* that are attached to one another at so-called *coupling points*, usually the joints connecting them. Such a subsystem can consist of a single rigid or flexible body or a compound of such. In industrial robots for example, it often occurs that a link of the robot arm is combined as a unit with the corresponding motor and gearbox. Therefore, it makes sense to combine these components into a subsystem, and thus to simplify the modeling of the robot. The kinematic topology of a subsystem can be arbitrary, so it can also contain closed kinematic loops.

2.1 Recursive kinematics

Instead of the generalized coordinates $\dot{\mathbf{q}}$, subsystems use the subsystem-specific velocity coordinates

$$\boldsymbol{V}_i = \begin{pmatrix} \mathbf{v}_{o,i} \\ \boldsymbol{\omega}_{o,i} \\ \dot{\mathbf{q}}_i \end{pmatrix}. \tag{5}$$

These consist of the guidance velocities $\mathbf{v}_{o,i}$ and $\boldsymbol{\omega}_{o,i}$ as well as the internal velocities $\dot{\mathbf{q}}_i$. The internal velocities describe the movement caused by components of the subsystem itself, while the guidance velocities introduce the motion of the system the current subsystem is coupled with. The first two components, $\mathbf{v}_{o,i}$ and $\boldsymbol{\omega}_{o,i}$, are the velocities of the preceding subsystem at the coupling point i, expressed in frame i, while the last one usually consists of joint velocities or velocities of flexible bodies as shown in Fig. 2.

The subsystem velocities can be calculated recursively through

$$\boldsymbol{V}_i = \mathbf{T}_{ip}\, \boldsymbol{V}_p + \mathbf{F}_i\, \dot{\mathbf{q}}_i, \tag{6}$$

with the index p indicating the parent subsystem of i, the velocity transformation matrix $\mathbf{T}_{ip}$ from p to i and $\mathbf{F}_i$ the matrix that maps the generalized velocities belonging to subsystem i to the internal velocities. Therefore, $\mathbf{F}_i$ usually is constant, like

$$\mathbf{F}_i = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{I} \end{pmatrix}. \tag{7}$$

Then the matrix $\mathbf{T}_{ip}$ transforms the velocities $\boldsymbol{V}_p$ of the preceding subsystem into the guidance velocity components $\mathbf{v}_{o,i}$ and $\boldsymbol{\omega}_{o,i}$, therefore it takes the shape

$$\mathbf{T}_{ip} = \begin{pmatrix} \mathbf{R}_{ip} & \mathbf{R}_{ip}\tilde{\mathbf{r}}_{pi}^{\top} & \frac{\partial \mathbf{v}_{pi}}{\partial \dot{\mathbf{q}}_p} \\ \mathbf{0} & \mathbf{R}_{ip} & \frac{\partial \boldsymbol{\omega}_{pi}}{\partial \dot{\mathbf{q}}_p} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{pmatrix}, \tag{8}$$

with $\mathbf{v}_{pi}$ and $\boldsymbol{\omega}_{pi}$ being the relative velocities between the coupling points p and i. The upper left 2x2-submatrix is a spatial transform as described in [23], Chap. 2.2. This matrix transforms the point at which the velocities $\mathbf{v}_{o,p}$ and $\boldsymbol{\omega}_{o,p}$ are defined from coupling point p to i while also changing the frame in which the velocities are described from p to i through the rotation matrix $\mathbf{R}_{ip}$. The ($\sim$)-operator is called the cross-product operator and turns a vector into a so-called cross-product matrix, so that $\mathbf{a} \times \mathbf{b} = \tilde{\mathbf{a}}\mathbf{b}$. The vector $\mathbf{r}_{pi}$ is the position vector connecting the coupling points of the subsystems p and i. The other component of $\mathbf{T}_{ip}$ is the Jacobian of the relative velocities $\mathbf{v}_{pi}$ and $\boldsymbol{\omega}_{pi}$ w.r.t. the internal velocities of p, represented in frame i.

This relationship between preceding and current subsystem velocities is called the *kinematic chain* [14] due to the chain-like structure of relationship (6). After evaluating this formula for all subsystems, the result can be rewritten as a global relationship

$$\boldsymbol{V} = \begin{pmatrix} \boldsymbol{V}_1 \\ \boldsymbol{V}_2 \\ \vdots \\ \boldsymbol{V}_N \end{pmatrix} = \begin{pmatrix} \mathbf{F}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{T}_{21}\mathbf{F}_1 & \mathbf{F}_2 & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{T}_{N1}\mathbf{F}_1 & \mathbf{T}_{N2}\mathbf{F}_2 & \cdots & \mathbf{F}_N \end{pmatrix} \begin{pmatrix} \dot{\mathbf{q}}_1 \\ \dot{\mathbf{q}}_2 \\ \vdots \\ \dot{\mathbf{q}}_N \end{pmatrix} = \mathbf{F}\dot{\mathbf{q}} \tag{9}$$

of the set of all subsystem velocities $\boldsymbol{V}_i$ and the generalized velocities $\dot{\mathbf{q}}$. Velocity transformation matrices over multiple subsystems, like $\mathbf{T}_{N1}$, are obtained by multiplication of neighboring transformation matrices, e.g., $\mathbf{T}_{ac} = \mathbf{T}_{ab}\mathbf{T}_{bc}$. For systems with a topological tree structure, $\mathbf{F}$ exhibits a lower triangular block structure, which will be exploited later on in the IIM.

2.2 Subsystem-based EoM

The dynamic equations of the subsystems (see Sect. A) give rise to the EoM of the multibody system

$$\sum_{i=1}^{N} \left(\frac{\partial \boldsymbol{V}_i}{\partial \dot{\mathbf{q}}} \right)^{\top} \left(\overline{\mathbf{M}}_i \dot{\boldsymbol{V}}_i + \overline{\mathbf{h}}_i(\boldsymbol{V}_i, \mathbf{q}, \mathbf{u}_i) \right) = \mathbf{0} \,. \tag{10}$$

The velocity vector $\boldsymbol{V}_i$ of subsystem i can be fully described. The matrix $\overline{\mathbf{M}}_i$ is the mass matrix of subsystem i and $\overline{\mathbf{h}}_i$ includes all the generalized subsystem forces apart from the inertia reaction forces. The transpose of the Jacobian $\frac{\partial \boldsymbol{V}_i}{\partial \dot{\mathbf{q}}}$, which is the row of $\mathbf{F}$, can be interpreted as the transformation from subsystem forces into the generalized forces. Further information on the composition of these subsystem matrices is compiled in appendix A.

Finally, the relation between the EoM in subsystem form (10) and the EoM in generalized coordinates (1b) is given obtained by substituting the relationships $\boldsymbol{V} = \mathbf{F}\dot{\mathbf{q}}$ and $\dot{\boldsymbol{V}} = \mathbf{F}\ddot{\mathbf{q}} + \dot{\mathbf{F}}\dot{\mathbf{q}}$, which results in

$$\mathbf{M}(\mathbf{q}) = \sum_{i=1}^{N} \left(\frac{\partial \boldsymbol{V}_i}{\partial \dot{\mathbf{q}}} \right)^{\top} \overline{\mathbf{M}}_i \frac{\partial \boldsymbol{V}_i}{\partial \dot{\mathbf{q}}} \tag{11}$$

$$\mathbf{h}(\mathbf{q},\dot{\mathbf{q}}) = \sum_{i=1}^{N} \left(\frac{\partial \boldsymbol{V}_i}{\partial \dot{\mathbf{q}}}\right)^{\top} \left[\overline{\mathbf{M}}_i \frac{\mathrm{d}}{\mathrm{d}\,t}\left(\frac{\partial \boldsymbol{V}_i}{\partial \dot{\mathbf{q}}}\right)\dot{\mathbf{q}} + \overline{\mathbf{h}}_i\right] \tag{12}$$

$$\mathbf{B}(\mathbf{q}) = \sum_{i=1}^{N} \left(\frac{\partial \boldsymbol{V}_i}{\partial \dot{\mathbf{q}}}\right)^{\top} \overline{\mathbf{B}}_i \tag{13}$$

with $\overline{\mathbf{B}}_i = -\frac{\partial \overline{\mathbf{h}}_i}{\partial \mathbf{u}}$. These global matrices now allow to solve the FD according to (2), i.e., the mass matrix $\mathbf{M}$ is inverted to obtain the system accelerations $\ddot{\mathbf{q}}$.

2.3 Subsystem $O(n)$ algorithm

The solution of the forward dynamics as shown in (2), however, has a computational complexity of $O(n^3)$ due to the necessary inversion of the mass matrix. For more efficient evaluation, various recursive algorithms with complexity $O(n)$ were proposed, e.g., [11–14]. Due to increased overhead, however, they only become faster than the direct solution of the global form for $n > 9$.

The $O(n)$-methods consist of three recursions. In the first recursion of the subsystem $O(n)$ algorithm [14], the kinematics is evaluated from the base subsystem outwards as in (6). Then, in the following inward recursion, the articulated-body inertia $\overline{\mathbf{M}}_i^*$ and bias forces $\overline{\mathbf{h}}_i^*$, as they are named in the ABA, are computed as

$$\overline{\mathbf{M}}_p^* = \overline{\mathbf{M}}_p + \sum_{i\in\{s(p)\}} \mathbf{T}_{ip}^{*\top} \overline{\mathbf{M}}_i^* \mathbf{T}_{ip} \tag{14}$$

$$\overline{\mathbf{h}}_p^* = \overline{\mathbf{h}}_p + \sum_{i\in\{s(p)\}} \mathbf{T}_{ip}^{*\top} \left(\overline{\mathbf{M}}_i^* \dot{\mathbf{T}}_{ip} \boldsymbol{V}_p + \overline{\mathbf{h}}_i^*\right), \tag{15}$$

$$\text{with} \quad \mathbf{T}_{ip}^* = (\mathbf{I} - \overline{\mathbf{M}}_i^* \mathbf{S}_i)^{\top} \mathbf{T}_{ip}, \tag{16}$$

$$\mathbf{S}_i = \mathbf{F}_i \overline{\mathbf{M}}_{Ri}^{-1} \mathbf{F}_i^{\top} \quad \text{and} \quad \overline{\mathbf{M}}_{Ri} = \mathbf{F}_i^{\top} \overline{\mathbf{M}}_i^* \mathbf{F}_i. \tag{17}$$

The set $\{s(p)\}$ consists of all subsystems directly attached to p, i.e., that have p as their common parent. Finally, the accelerations are calculated through the final outward recursion

$$\ddot{\mathbf{q}}_i = -\overline{\mathbf{M}}_{Ri}^{-1} \mathbf{F}_i^{\top} \left[\overline{\mathbf{M}}_i^* (\mathbf{T}_{ip} \dot{\boldsymbol{V}}_p + \dot{\mathbf{T}}_{ip} \boldsymbol{V}_p) + \overline{\mathbf{h}}_i^*\right] \tag{18}$$

$$\dot{\boldsymbol{V}}_i = \mathbf{T}_{ip} \dot{\boldsymbol{V}}_p + \dot{\mathbf{T}}_{ip} \boldsymbol{V}_p + \mathbf{F}_i \ddot{\mathbf{q}}_i + \dot{\mathbf{F}}_i \dot{\mathbf{q}}_i. \tag{19}$$

As can be seen from (17) and (18), the only matrix that needs to be inverted here is $\overline{\mathbf{M}}_{Ri}$, whose dimension is that of the internal velocities of the respective subsystem. The downside of this method is, however, that it does not allow the splitting of the accelerations as in (2). Therefore, a different method is needed to calculate the IWIM.

3 The extended unit force method (eUFM)

One method to calculate the inverse OSIM that fulfils the criteria mentioned in Sect. 1 is shown in [20]. It builds on the unit force method of [16] but is augmented so that it does not rely on additional assumptions to simplify the system dynamics. Because of this, it will be called the extended unit force method.

To calculate the inverse OSIM $\mathbf{\Lambda}^{-1} = \mathbf{J}_t \mathbf{M}^{-1} \mathbf{J}_t^\top$, the matrix $\mathbf{M}^{-1}\mathbf{J}_t^\top$ is evaluated first since this term can be taken directly from the FD

$$\ddot{\mathbf{q}} = -\mathbf{M}^{-1}\left(\mathbf{h}(\mathbf{q}, \dot{\mathbf{q}}) - \mathbf{J}_a^\top \boldsymbol{\tau}\right) + \mathbf{M}^{-1}\mathbf{J}_t^\top \mathbf{F}_c. \tag{20}$$

This can also be written as $\ddot{\mathbf{q}} = \ddot{\mathbf{q}}_{[0]} + \mathbf{M}^{-1}\mathbf{J}_t^\top \mathbf{F}_c$, where $\ddot{\mathbf{q}}_{[0]}$ are the system accelerations according to the current state and actuator torques, which can be obtained through the $O(n)$-method. The matrix $\mathbf{M}^{-1}\mathbf{J}_t^\top$ can then be calculated column-wise by choosing a unit vector $\mathbf{e}_i$ (whose ith entry is 1) as force input. When solving for the accelerations, the result is

$$\ddot{\mathbf{q}}_{[i]} = \ddot{\mathbf{q}}_{[0]} + \mathbf{M}^{-1}\mathbf{J}_t^\top \mathbf{e}_i, \tag{21}$$

so $\ddot{\mathbf{q}}_{[i]}$ are the system accelerations for input $u_i = 1$. Now, the ith column of $\mathbf{M}^{-1}\mathbf{J}_t^\top$ is simply the difference $\ddot{\mathbf{q}}_{[i]} - \ddot{\mathbf{q}}_{[0]}$. By repeating this process for each element of the contact force vector, the entire matrix is obtained. To get the inverse OSIM now, this result is premultiplied by $\mathbf{J}_t$. This final multiplication can just be left out as mentioned in Sect. 1. Now, the input for the eUFM can be extended to $\mathbf{u} = \begin{pmatrix}\boldsymbol{\tau}^\top & \mathbf{F}_c^\top\end{pmatrix}^\top$ so that the dynamics can now be expressed as in (2) with $\mathbf{B} = \begin{bmatrix}\mathbf{J}_a^\top & \mathbf{J}_t^\top\end{bmatrix}$. For this case, the eUFM now returns $\hat{\mathbf{B}} = \begin{bmatrix}\mathbf{M}^{-1}\mathbf{J}_a^\top & \mathbf{M}^{-1}\mathbf{J}_t^\top\end{bmatrix}$ instead of $\mathbf{M}^{-1}\mathbf{J}_t^\top$.

The drawback of this method is that the $O(n)$-algorithm has to be evaluated $m+1$ times for $\mathbf{u} \in \mathbb{R}^m$, resulting in a computational complexity of $O(m\,n)$. This reduces the usability of this method for systems with higher numbers of inputs.

4 The implicit inversion method (IIM)

This section presents the main contribution of this paper, the implicit inversion method. It is a way to evaluate the IWIM without having to explicitly invert the global mass matrix, hence the name. The aim is to derive an algorithm from the subsystem $O(n)$-method that is computationally less expensive than those presented in the previous sections. The algorithm should be efficient in particular for systems with multiple inputs while still working with subsystem quantities as far as possible. As mentioned in the introduction, we only consider systems that exhibit a linear relation between inputs and accelerations. Thus the $\overline{\mathbf{h}}_i$ in (10) are linear in the inputs $\mathbf{u}$ and can be partitioning into

$$\overline{\mathbf{h}}_i = \overline{\mathbf{h}}_{0,i} + \overline{\mathbf{h}}_{u,i} \qquad \text{with} \qquad \overline{\mathbf{h}}_{u,i} = -\overline{\mathbf{B}}_i \mathbf{u}\,, \tag{22}$$

where the matrices $\overline{\mathbf{B}}_i$ determine how the inputs $\mathbf{u}$ affect the subsystem. The vector $\overline{\mathbf{h}}_{0,i}$ contains the sum of all subsystem forces that do not depend on the inputs $\mathbf{u}$. The FD is evaluated using the $O(n)$-method presented in Sect. 2.3 so that most of the terms needed for the IIM are already available.

4.1 Basic form of the IIM

4.1.1 Computing the input-based accelerations $\ddot{\mathbf{q}}_u$

The derivation of the IWIM is done by going through the recursive FD method step by step and picking out all the terms that are related to the inputs. That means that the first step of the $O(n)$-method, the evaluation of the kinematic chain can be skipped, as it is not affected by the inputs.

In the second step, the bias forces do depend on the inputs. Substituting (22) in the propagation formula for the bias forces (15) gives

$$\begin{aligned} \overline{\mathbf{h}}_p^* &= \overline{\mathbf{h}}_{0,p}^* + \overline{\mathbf{h}}_{u,p}^* \\ \text{with} \quad \overline{\mathbf{h}}_{u,p}^* &= \overline{\mathbf{h}}_{u,p} + \sum_{i \in \{s(p)\}} \mathbf{T}_{ip}^{*\top} \overline{\mathbf{h}}_{u,i}^* = -\overline{\mathbf{B}}_p \mathbf{u} + \sum_{i \in \{s(p)\}} \mathbf{T}_{ip}^{*\top} \overline{\mathbf{h}}_{u,i}^* \,, \end{aligned} \tag{23}$$

a recursive formula for the input-dependent component of the bias forces $\overline{\mathbf{h}}_{u,i}^*$. Rewriting this formula to a global form leads to the relationship

$$\overline{\mathbf{h}}_u^* = -\mathbf{T}^{*\top} \overline{\mathbf{B}} \mathbf{u} \tag{24}$$

with

$$\mathbf{T}^* = \begin{pmatrix} \mathbf{I} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{T}_{21}^* & \mathbf{I} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{T}_{N1}^* & \mathbf{T}_{N2}^* & \cdots & \mathbf{I} \end{pmatrix}. \tag{25}$$

The matrices $\mathbf{T}_{ac}^*$ propagate as $\mathbf{T}_{ac}^* = \mathbf{T}_{ab}^* \mathbf{T}_{bc}^*$.

In the third step of the $O(n)$-method, the accelerations are obtained through a final outward recursion (18). Here, the variables $\dot{\boldsymbol{V}}_i$ and $\overline{\mathbf{h}}_i^*$ depend on $\mathbf{u}$. To separate accelerations from inputs, a change to global variables is necessary for the input-dependent accelerations $\ddot{\mathbf{q}}_u$. So, inserting (2) into $\dot{\boldsymbol{V}}_i = \frac{\partial \boldsymbol{V}_i}{\partial \dot{\mathbf{q}}} \ddot{\mathbf{q}} + \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial \boldsymbol{V}_i}{\partial \dot{\mathbf{q}}}\right) \dot{\mathbf{q}}$ leads to

$$\dot{\boldsymbol{V}}_i = \dot{\boldsymbol{V}}_{0,i} + \left(\frac{\partial \boldsymbol{V}_i}{\partial \dot{\mathbf{q}}}\right) \ddot{\mathbf{q}}_u \qquad \text{with} \qquad \dot{\boldsymbol{V}}_{0,i} = \frac{\partial \boldsymbol{V}_i}{\partial \dot{\mathbf{q}}} \ddot{\mathbf{q}}_0 + \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial \boldsymbol{V}_i}{\partial \dot{\mathbf{q}}}\right) \dot{\mathbf{q}} \,. \tag{26}$$

That way, all state-dependent accelerations can still be calculated as before while extracting the calculation of $\ddot{\mathbf{q}}_{u,i}$ as

$$\ddot{\mathbf{q}}_{u,i} = -\overline{\mathbf{M}}_{Ri}^{-1} \mathbf{F}_i^\top \left[\overline{\mathbf{M}}_i^* \mathbf{T}_{ip} \left(\frac{\partial \boldsymbol{V}_p}{\partial \dot{\mathbf{q}}}\right) \ddot{\mathbf{q}}_u + \overline{\mathbf{h}}_{u,i}^* \right]. \tag{27}$$

To avoid mixing variables on subsystem level and global level, namely $\ddot{\mathbf{q}}_{u,i}$ and $\ddot{\mathbf{q}}_u$, the whole equation is expanded to the global form

$$\ddot{\mathbf{q}}_u = -\mathrm{diag}\left(\overline{\mathbf{M}}_{Ri}^{-1} \mathbf{F}_i^\top \overline{\mathbf{M}}_i^*\right) (\mathbf{F} - \mathrm{diag}(\mathbf{F}_i)) \ddot{\mathbf{q}}_u - \mathrm{diag}\left(\overline{\mathbf{M}}_{Ri}^{-1} \mathbf{F}_i^\top\right) \overline{\mathbf{h}}_u^* \,. \tag{28}$$

Substituting (24), it can be further simplified to

$$\left[\mathbf{I} + \mathrm{diag}\left(\overline{\mathbf{M}}_{Ri}^{-1} \mathbf{F}_i^\top \overline{\mathbf{M}}_i^*\right) (\mathbf{F} - \mathrm{diag}(\mathbf{F}_i))\right] \ddot{\mathbf{q}}_u = \mathrm{diag}\left(\overline{\mathbf{M}}_{Ri}^{-1} \mathbf{F}_i^\top\right) \mathbf{T}^{*\top} \overline{\mathbf{B}} \mathbf{u} \,, \tag{29}$$

and using (17), the expression $\mathbf{I} - \mathrm{diag}\left(\overline{\mathbf{M}}_{Ri}^{-1} \mathbf{F}_i^\top \overline{\mathbf{M}}_i^* \mathbf{F}_i\right)$ vanishes and the equation is reduced to

$$\left[\mathrm{diag}\left(\overline{\mathbf{M}}_{Ri}^{-1} \mathbf{F}_i^\top \overline{\mathbf{M}}_i^*\right) \mathbf{F}\right] \ddot{\mathbf{q}}_u = \mathrm{diag}\left(\overline{\mathbf{M}}_{Ri}^{-1} \mathbf{F}_i^\top\right) \mathbf{T}^{*\top} \overline{\mathbf{B}} \mathbf{u} \,. \tag{30}$$

On the right-hand side of this equation, the product $\mathbf{T}^* \mathrm{diag}\,(\mathbf{F}_i)$ takes the shape

$$\mathbf{T}^* \mathrm{diag}\,(\mathbf{F}_i) = \begin{pmatrix} \mathbf{F}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{T}^*_{21}\mathbf{F}_1 & \mathbf{F}_2 & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{T}^*_{N1}\mathbf{F}_1 & \mathbf{T}^*_{N2}\mathbf{F}_2 & \cdots & \mathbf{F}_N \end{pmatrix}, \tag{31}$$

which looks like $\mathbf{F}$, the only difference being that all transformations $\mathbf{T}_{ip}$ are replaced by $\mathbf{T}^*_{ip}$. Therefore, the product is summarized as $\mathbf{F}^* = \mathbf{T}^* \mathrm{diag}\,(\mathbf{F}_i)$. (30) also shows an acceleration transformation matrix $\boldsymbol{\Gamma} = \mathrm{diag}\left(\overline{\mathbf{M}}_{Ri}^{-1}\mathbf{F}_i^\top \overline{\mathbf{M}}_i^*\right)\mathbf{F}$ that needs to be inverted to obtain the input-dependent accelerations

$$\ddot{\mathbf{q}}_u = \boldsymbol{\Gamma}^{-1}\mathrm{diag}\left(\overline{\mathbf{M}}_{Ri}^{-1}\right)\mathbf{F}^{*\top}\overline{\mathbf{B}}\mathbf{u}\,. \tag{32}$$

4.1.2 Inversion of $\boldsymbol{\Gamma}$

The matrix $\boldsymbol{\Gamma}$ retains the structure of $\mathbf{F}$, i.e., it still is a lower triangular block matrix. Besides, thanks to $\overline{\mathbf{M}}_{Ri}^{-1}\mathbf{F}_i^\top \overline{\mathbf{M}}_i^* \mathbf{F}_i = \mathbf{I}$, all the block matrices on the diagonal of $\boldsymbol{\Gamma}$ are identity matrices, leading to a lower unitriangular structure, i.e., a lower triangular matrix with only ones as its diagonal entries. This guarantees that $\boldsymbol{\Gamma}$ will always be invertible.

Using this special structure, the inversion of $\boldsymbol{\Gamma}$ can be done through the cascaded inversion of its submatrices as shown in the following example for a 3-link chain.

$$\boldsymbol{\Gamma} = \begin{pmatrix} \overset{\mathbf{A}_1}{\mathbf{I}} & \overset{\mathbf{A}_2}{0} & \overset{\mathbf{A}_3}{0} \\ \cdot & \mathbf{I} & 0 \\ \cdot & \cdot & \mathbf{I} \end{pmatrix}. \tag{33}$$

The inverse of the resulting block matrix form

$$\mathbf{A}_i = \begin{pmatrix} \mathbf{A}_p & \mathbf{0} \\ \mathbf{C}_i & \mathbf{I} \end{pmatrix} \tag{34}$$

is

$$\mathbf{A}_i^{-1} = \begin{pmatrix} \mathbf{A}_p^{-1} & \mathbf{0} \\ -\mathbf{C}_i\mathbf{A}_p^{-1} & \mathbf{I} \end{pmatrix} \tag{35}$$

for quadratic diagonal block matrices, which is guaranteed for these matrices. By starting this inversion process with the upper left submatrix $\mathbf{A}_1 = \mathbf{I}$ and then iteratively expanding until $\mathbf{A}_N = \boldsymbol{\Gamma}$, the inverse of the original matrix is obtained without any explicit matrix inversion, which motivates to call this approach the *implicit inversion method* (IIM).

4.2 Subsystem-based inversion of $\boldsymbol{\Gamma}$

The terms that appear after following the sequential approach of building the inverse can be further simplified once more, leading to a closed form solution for the inverse. This is

shown with the help of an example here, but the result can be generalized to all systems with a topological chain or tree structure.

To show the general structure of the solution, a minimum of three consecutive links is necessary. That is why a 3-link pendulum will serve as an example here. The kinematic Jacobian from (9) for this system has the shape of

$$\mathbf{F} = \begin{pmatrix} \mathbf{F}_1 & \mathbf{0} & \mathbf{0} \\ \mathbf{T}_{21}\mathbf{F}_1 & \mathbf{F}_2 & \mathbf{0} \\ \mathbf{T}_{31}\mathbf{F}_1 & \mathbf{T}_{32}\mathbf{F}_2 & \mathbf{F}_3 \end{pmatrix}, \tag{36}$$

which, when following the notation of (34) and introducing $\mathbf{C}_i = [\mathbf{c}_{ij}]_{j=1..p}$, leads to

$$\mathbf{\Gamma} = \operatorname{diag}(\overline{\mathbf{M}}_{Ri}^{-1}\mathbf{F}_i^\top \overline{\mathbf{M}}_i^*)\mathbf{F} = \begin{pmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{c}_{21} & \mathbf{I} & \mathbf{0} \\ \mathbf{c}_{31} & \mathbf{c}_{32} & \mathbf{I} \end{pmatrix} \tag{37}$$

with

$$\begin{aligned} \mathbf{c}_{21} &= \overline{\mathbf{M}}_{R2}^{-1}\mathbf{F}_2^\top \overline{\mathbf{M}}_2^* \mathbf{T}_{21}\mathbf{F}_1, \\ \mathbf{c}_{31} &= \overline{\mathbf{M}}_{R3}^{-1}\mathbf{F}_3^\top \overline{\mathbf{M}}_3^* \mathbf{T}_{31}\mathbf{F}_1, \\ \mathbf{c}_{32} &= \overline{\mathbf{M}}_{R3}^{-1}\mathbf{F}_3^\top \overline{\mathbf{M}}_3^* \mathbf{T}_{32}\mathbf{F}_2 \,. \end{aligned} \tag{38}$$

Following (34), we get the submatrices

$$\mathbf{A}_1 = \mathbf{I}, \qquad \mathbf{A}_2 = \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{c}_{21} & \mathbf{I} \end{pmatrix}, \qquad \mathbf{A}_3 = \mathbf{\Gamma}\,, \tag{39}$$

for which (35) gives the inverses

$$\mathbf{A}_1^{-1} = \mathbf{I}, \qquad \mathbf{A}_2^{-1} = \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ -\mathbf{c}_{21} & \mathbf{I} \end{pmatrix}, \qquad \mathbf{A}_3^{-1} = \begin{pmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} \\ -\mathbf{c}_{21} & \mathbf{I} & \mathbf{0} \\ -\mathbf{c}_{31} + \mathbf{c}_{32}\mathbf{c}_{21} & -\mathbf{c}_{32} & \mathbf{I} \end{pmatrix}. \tag{40}$$

In the matrix $\mathbf{A}_3^{-1}$, the lower left element is the only entry that is not explicitly known yet. Inserting (38) into (40), we get

$$-\mathbf{c}_{31} + \mathbf{c}_{32}\mathbf{c}_{21} = -\overline{\mathbf{M}}_{R3}^{-1}\mathbf{F}_3^\top \overline{\mathbf{M}}_3^* (\mathbf{T}_{31} - \mathbf{T}_{32}\mathbf{F}_2 \overline{\mathbf{M}}_{R2}^{-1}\mathbf{F}_2^\top \overline{\mathbf{M}}_2^* \mathbf{T}_{21})\mathbf{F}_1\,. \tag{41}$$

Using the relationships $\mathbf{T}_{31} = \mathbf{T}_{32}\mathbf{T}_{21}$ and $\mathbf{S}_i = \mathbf{F}_i \overline{\mathbf{M}}_{Ri}^{-1}\mathbf{F}_i^\top$, the result can be simplified further to

$$-\mathbf{c}_{31} + \mathbf{c}_{32}\mathbf{c}_{21} = -\overline{\mathbf{M}}_{R3}^{-1}\mathbf{F}_3^\top \overline{\mathbf{M}}_3^* \mathbf{T}_{32}(\mathbf{I} - \mathbf{S}_2\overline{\mathbf{M}}_2^*)\mathbf{T}_{21}\mathbf{F}_1\,, \tag{42}$$

which can be further reduced by using the properties $\mathbf{S}_i = \mathbf{S}_i^\top$ and $\overline{\mathbf{M}}_i^* = \overline{\mathbf{M}}_i^{*\top}$. These imply $(\mathbf{I} - \mathbf{S}_i\overline{\mathbf{M}}_i^*) = (\mathbf{I} - \overline{\mathbf{M}}_i^*\mathbf{S}_i)^\top$, which allows for the substitution (17). This gives us the final result of

$$-\mathbf{c}_{31} + \mathbf{c}_{32}\mathbf{c}_{21} = -\overline{\mathbf{M}}_{R3}^{-1}\mathbf{F}_3^\top \overline{\mathbf{M}}_3^* \mathbf{T}_{32}\mathbf{T}_{21}^* \mathbf{F}_1 \tag{43}$$

for this entry. With that, the matrix

$$\mathbf{A}_3^{-1} = \begin{pmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} \\ -\overline{\mathbf{M}}_{R2}^{-1}\mathbf{F}_2^{\top}\overline{\mathbf{M}}_2^{*}\mathbf{T}_{21}\mathbf{F}_1 & \mathbf{I} & \mathbf{0} \\ -\overline{\mathbf{M}}_{R3}^{-1}\mathbf{F}_3^{\top}\overline{\mathbf{M}}_3^{*}\mathbf{T}_{32}\mathbf{T}_{21}^{*}\mathbf{F}_1 & -\overline{\mathbf{M}}_{R3}^{-1}\mathbf{F}_3^{\top}\overline{\mathbf{M}}_3^{*}\mathbf{T}_{32}\mathbf{F}_2 & \mathbf{I} \end{pmatrix} \tag{44}$$

is the result of the inversion procedure, which can also be written as

$$\mathbf{A}_3^{-1} = \mathbf{I} - \mathrm{diag}(\overline{\mathbf{M}}_{Ri}^{-1}\mathbf{F}_i^{\top}\overline{\mathbf{M}}_i^{*}\mathbf{T}_{ip})\hat{\mathbf{F}}\,. \tag{45}$$

In (45), only a single variable necessary for the calculation of the inverse is new compared to $\mathbf{\Gamma}$, namely

$$\hat{\mathbf{F}} = \begin{pmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{F}_1 & \mathbf{0} & \mathbf{0} \\ \mathbf{T}_{21}^{*}\mathbf{F}_1 & \mathbf{F}_2 & \mathbf{0} \end{pmatrix}\,. \tag{46}$$

When comparing $\hat{\mathbf{F}}$ to (31), it can be seen that each row i in $\hat{\mathbf{F}}$ corresponds to the row of its predecessor $p(i)$ in $\mathbf{F}^{*}$. With this, the general form of the matrix $\hat{\mathbf{F}}$ can be derived as

$$\hat{\mathbf{F}} = [\,\mathbf{f}_i\,]_{i=1..N} \qquad \text{with} \qquad \mathbf{f}_i = \begin{cases} \mathbf{0} & p(i) = 0 \\ \mathbf{F}_p^{*} & p(i) > 0 \end{cases} \tag{47}$$

where $p(i)$ denotes the predecessor of i and $\mathbf{F}_p^{*}$ the rows of $\mathbf{F}^{*}$ belonging to subsystem p. From this, a general rule for the calculation of $\mathbf{\Gamma}^{-1}$ can be deduced, namely

$$\mathbf{\Gamma}^{-1} = \mathbf{I} - \mathrm{diag}(\overline{\mathbf{M}}_{Ri}^{-1}\mathbf{F}_i^{\top}\overline{\mathbf{M}}_i^{*}\mathbf{T}_{ip})\hat{\mathbf{F}}. \tag{48}$$

4.3 Recursive computation of the IIM

Inserting the inverse (48) into the previous formula (32) for the input-based accelerations gives the final result

$$\ddot{\mathbf{q}}_u = [\mathbf{I} - \mathrm{diag}(\overline{\mathbf{M}}_{Ri}^{-1}\mathbf{F}_i^{\top}\overline{\mathbf{M}}_i^{*}\mathbf{T}_{ip})\hat{\mathbf{F}}]\,\mathrm{diag}(\overline{\mathbf{M}}_{Ri}^{-1})\mathbf{F}^{*\top}\overline{\mathbf{B}}\mathbf{u}\,. \tag{49}$$

In this formula, the only inverses remaining are those of $\overline{\mathbf{M}}_{Ri}$, which were already calculated during the execution of the $O(n)$-method, so no separate matrix inversions are needed. This yields $\ddot{\mathbf{q}} = \ddot{\mathbf{q}}_0 + \hat{\mathbf{B}}\mathbf{u}$ with $\ddot{\mathbf{q}}_0$ obtained by running the $O(n)$-method without any inputs as mentioned before. During this evaluation, all variables that are necessary to calculate the global matrices $\mathbf{F}^{*}$, therefore $\hat{\mathbf{F}}$ and subsequently $\hat{\mathbf{B}}$, are computed in advance. Finally, the global form of the IWIM $\hat{\mathbf{B}}$ is obtained as

$$\hat{\mathbf{B}} = [\mathbf{I} - \mathrm{diag}(\overline{\mathbf{M}}_{Ri}^{-1}\mathbf{F}_i^{\top}\overline{\mathbf{M}}_i^{*}\mathbf{T}_{ip})\hat{\mathbf{F}}]\,\mathrm{diag}(\overline{\mathbf{M}}_{Ri}^{-1})\mathbf{F}^{*\top}\overline{\mathbf{B}}. \tag{50}$$

However, the local nature of the resulting matrices also allows for a recursive reformulation. The fact that $\hat{\mathbf{F}}$ only consists of components from $\mathbf{F}^{*}$ can be utilized to further reduce the number of variables. The result is summarized in algorithm 1. The bracketed indices refer to all rows/columns corresponding to the respective subsystem. If two indices are separated by a colon, e.g., $[p:i]$, this refers to all columns from p to i. That way, the sparsity patterns introduced by the topology of the system through $\mathbf{F}$ are utilized to accelerate the calculation of the IWIM.

Algorithm 1 Pseudo-code for the IIM

Step 1: Evaluate the $O(n)$-method according to Sect. 2.3 without inputs
$\ddot{\mathbf{q}}_0 = \mathbf{f}(\mathbf{q}, \dot{\mathbf{q}})$

Step 2: Outward recursion to calculate $\mathbf{F}^*$ and $\mathbf{\Gamma}^{-1}$.
for $i = 1 \rightarrow N$ **do**
  $\mathbf{F}^*_{[i,i]} = \mathbf{F}_i$
  $\mathbf{\Gamma}^{-1}_{[i,i]} = \mathbf{I}$
  if p exists **then**
    $\mathbf{F}^*_{[i,1:p]} = \mathbf{T}^*_{ip}\,\mathbf{F}^*_{[p,1:p]}$
    $\mathbf{\Gamma}^{-1}_{[i,1:p]} = -\mathbf{M}^{-1}_{Ri}\mathbf{F}^{\top}_i\mathbf{M}^*_i\mathbf{T}_{ip}\mathbf{F}^*_{[p,1:p]}$
  end if
end for

Step 3: Inward recursion to get $\hat{\mathbf{B}}$
for $i = N \rightarrow 1$ **do**
  $\hat{\mathbf{B}}_{[i:N,:]} = \hat{\mathbf{B}}_{[i:N,:]} + \mathbf{\Gamma}^{-1}_{[i:N,i]}\mathbf{M}^{-1}_{Ri}\mathbf{F}^{*\top}_{[i:N,i]}\overline{\mathbf{B}}_{[i:N,:]}$
end for

5 Simulation results

To analyze the performance of the proposed method, the effect of different topologies, numbers of DOF, and inputs is studied through simulation. The runtime of the three methods to calculate the IWIM presented in this paper will be compared here. These are the calculation of the IWIM by inversion of the global mass matrix as mentioned in Sect. 1 ,[1] the eUFM from Sect. 3, and the IIM as shown in algorithm 1. For that, the algorithms are implemented in Matlab 2023b and run on an Apple M2 Pro processor. The runtime is taken as the average evaluation time for the FD of a model over 1000 runs with randomized values for the states and inputs for each run.

5.1 n-link chain

At first, we look at systems with a pure chain structure, i.e., the model is an n-link pendulum with n ranging from 1 to 50. For this model, all subsystems were assumed to be a single rigid body with identical parameters. In Fig. 3 (a), only the last link is actuated, i.e., the number of inputs remained constant over the changing number of DOF.

In case only one joint is actuated, all the methods perform very similarly up until $\sim$ 18 DOF. Above that, the global inverse form starts to get noticeably slower, and up to at least 50 DOF, the IIM outperforms the eUFM marginally. From this, it can be seen that the execution of algorithm 1 for models of this scale is faster than the evaluation of the $O(n)$ method, as that is what the eUFM does in this case. Systems with only a single input are a very rare case, however, so the same n-link pendulum is modeled as a fully actuated system next. The results for these simulations can be seen in Fig. 3 (b). When looking at the results for the eUFM now, it becomes apparent why this method should not be used for systems with multiple inputs, since the resulting computational cost becomes prohibitive. Comparing the IIM with the global inverse form, the runtime for these methods does not

[1] i.e., the calculation of $\hat{\mathbf{B}} = \mathbf{M}^{-1}\mathbf{B}$ and $\ddot{\mathbf{q}}_0 = -\mathbf{M}^{-1}\mathbf{h}$ using a Cholesky factorization.

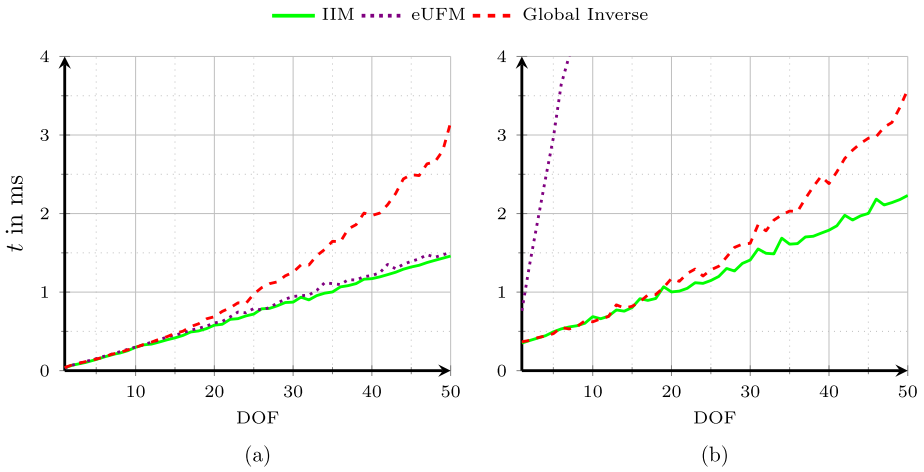

Fig. 3 Run time comparisons for the forward dynamics evaluation n-link pendulum models with (a) a single input and (b) the fully actuated models

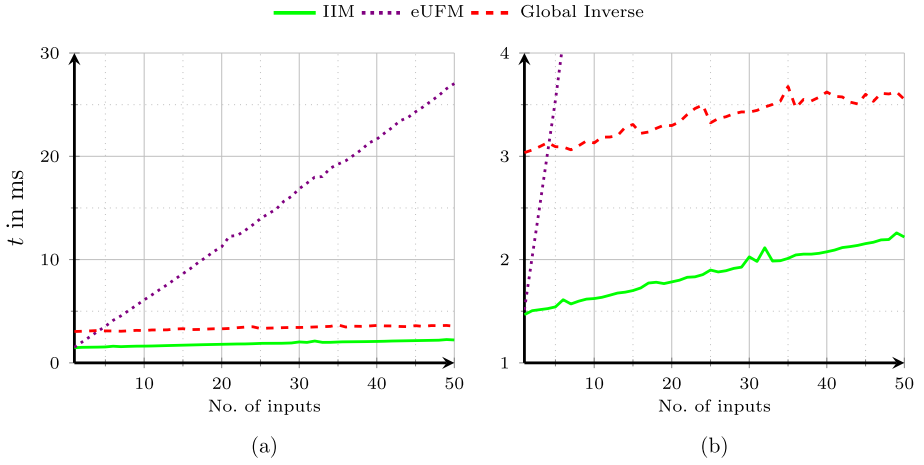

Fig. 4 Runtime comparison for a 50-link pendulum with a varying number of inputs (a) shows the full results and (b) is zoomed in to show the behavior of the IIM and the global inverse

change much compared to the single input case, the point where the IIM becomes faster is still at around $\approx$ 18-19 DOF.

To get a better understanding of the influence the number of inputs has on runtime, the 50-link pendulum model was simulated with a varying number of actuated links. The results of which are shown in Fig. 4 (a). Here, we can see a steep linear increase for the eUFM for an increasing number of inputs as expected. The increase in runtime for both the global inverse and the IIM is also linear, albeit much slower. It is necessary to zoom in to Fig. 4 (b), to get a better picture of the increase for these methods.

Table 1 Average runtimes for a 50-link pendulum

	Single input	Fully actuated	
Global Form	3.03	3.55	ms
IIM	1.46	2.21	ms
eUFM	1.51	27.04	ms

Table 2 Average runtimes for a 50-link binary tree

	1 input	50 inputs	
Global Form	2.77	3.49	ms
IIM	1.46	2.31	ms
eUFM	1.51	27.13	ms

When comparing the extreme cases of Fig. 4, the values of which are listed in Table 1, the increase of computational time per additional input is at $\approx 1\%$ for the IIM and $\approx 0.33\%$ for the global inverse form, while for the eUFM, it rises by $\approx 33\%$ per input.

The low impact the number of inputs has on the IIM can be explained when looking at (50). Here, the inputs only directly affect $\overline{\mathbf{B}}$, which is very sparsely populated, reducing the computational effort. This reasoning also applies to the solution in global inverse form since it uses $\overline{\mathbf{B}}$ as well.

5.2 n-element binary tree

Opposed to the n-link chain, which only has a single branch in its topological graph, the next test model is an n-link complete binary tree, which has the highest number of branching points possible. Although this structure makes no sense for practical systems, all systems with topological chain or tree structure are somewhere in between those two models. Therefore, the runtime for an arbitrary system should lie somewhere between these two test models. For that, the same tests were then conducted on models with a complete binary tree for their topological graph to check how the different system topology affects the calculation time.

The runtimes for the solution of the global inverse decrease slightly, while the other methods are not really impacted by the change in topology as can be seen in Table 2. The fact that the runtime for the IIM does not change is because the sparsity pattern left by tree structures is not yet exploited in the current implementation of the IIM.

All in all, it seems as if the system topology does not significantly affect the speed of these methods, just the number of DOF and in the case of the eUFM, the inputs.

5.3 Application example: a bipedal robot

Finally, the three methods are tested on a more practical application, a bipedal robot with 20 DOF and 38 inputs as shown in Fig. 1. The system consists of 14 motor-arm subsystems as mentioned in Sect. 2.2 and of one floating base subsystem for the robot torso. The large number of inputs is due to the number of contact points on the robot. There are 8 contact points defined for the robot with 4 per foot, so there are 24 contact forces and 14 motor torques.

For this system, the FD is calculated for a set of 1000 random states and inputs. The results of these simulations are shown in Table 3, and they show the same trends as were

Table 3 Average computational times for the bipedal robot

	Humanoid	
Global Form	0.67	ms
IIM	0.62	ms
eUFM	6.37	ms

observed in the n-link pendulum and tree. Since this system has a fairly large number of inputs, the eUFM is obviously the slowest of the three methods. For the other two methods, the results also make sense since the system has only 20 DOF, the IIM is only slightly faster than the solution of the inverse when compared with the results obtained for the chain and tree models.

6 Conclusions

This contribution introduces the *implicit inversion method* (IIM), a recursive method that leverages the subsystem formulation to effectively calculate the *inertia-weighted input matrix* (IWIM) $\hat{\mathbf{B}} = \mathbf{M}^{-1}\mathbf{B}$, which is the Jacobian of the *forward dynamics* (FD) w.r.t. the system inputs. The IIM is used in conjunction with an $O(n)$-algorithm to evaluate the state-dependent dynamics and makes use of intermediate results coming from the evaluation of the algorithm. Therefore, the IWIM can be calculated without having knowledge of the mass matrix $\mathbf{M}$ of the system in its generalized coordinate form. Both the accelerations from the O(n)-algorithm and the IWIM can then be used to solve an optimization of the form (4). Using all the byproducts of these methods additionally contributes to the efficiency of this method.

To determine the performance of this method, runtime comparisons were made between the IIM, an extended form of the unit force method, and the direct calculation of the IWIM through the inverse of the joint-space mass matrix using a Cholesky factorization. The IIM generally performs at least at the level of the other methods, and for systems above ~ 18 DOF and multiple inputs outperforms them increasingly.

However, it still needs to be compared against methods that are more efficient in calculating the inverse mass matrix, such as the one mentioned in [15] to get a better understanding of its standing. In theory, they should perform very similarly, since they are structurally very close to one another.

For a fair comparison though, the method should be reimplemented in a different programming language, e.g., in C++ instead of the current implementation, which is done in Matlab.

Appendix A: Subsystem dynamics

As shown in Fig. 5, subsystem i consists of N_{sub} bodies. The mass matrix of body j expressed in a body-fixed frame located at the center of mass is

$$\mathcal{M}_j = \mathrm{diag}(m_j\mathbf{I}, \boldsymbol{\Theta}_j^{C_j}), \tag{A1}$$

where $\boldsymbol{\Theta}_j^{C_j}$ is the inertia tensor and m_j the mass of body j. Each body acts a spatial wrench $\mathcal{W}_j$, which is expressed in frame j and accounts for gravity, friction, actuation, and other

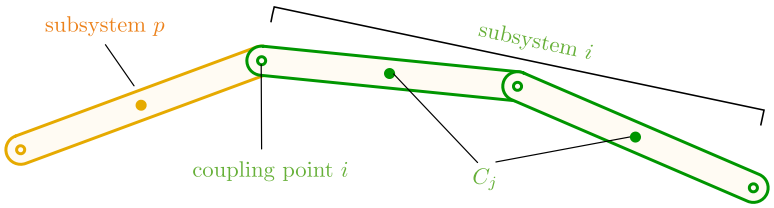

Fig. 5 Example of a subsystem i consisting of two bodies $j = 1, 2$

external loads. The Newton–Euler equations of body j expressed in the body-fixed frame are then written as

$$\mathcal{M}_j \dot{\mathbf{V}}_{C_j} + \widetilde{\mathbf{\Omega}}_j \mathcal{M}_j \mathbf{V}_{C_j} - \mathcal{W}_j = \mathbf{0}, \tag{A2}$$

where the matrix $\widetilde{\mathbf{\Omega}}_j = \mathrm{diag}(\widetilde{\boldsymbol{\omega}}_j, \widetilde{\boldsymbol{\omega}}_j)$ contains the cross-product matrix of the angular velocity of j. The twist $\mathbf{V}_{C_j}$ of body j is related to the subsystem velocity $\boldsymbol{V}_i$ by

$$\mathbf{V}_{C_j} = \frac{\partial \mathbf{V}_{C_j}}{\partial \boldsymbol{V}_i} \boldsymbol{V}_i. \tag{A3}$$

Introducing this into (A2), the principle of virtual power yields the dynamic equations of subsystem i

$$\overline{\mathbf{M}}_i \dot{\boldsymbol{V}}_i + \overline{\mathbf{G}}_i \boldsymbol{V}_i - \overline{\mathbf{Q}}_i = \mathbf{0} \tag{A4}$$

with

$$\overline{\mathbf{M}}_i = \sum_{j=1}^{N_{sub}} \left(\frac{\partial \mathbf{V}_{C_j}}{\partial \boldsymbol{V}_i} \right)^\top \mathcal{M}_j \frac{\partial \mathbf{V}_{C_j}}{\partial \boldsymbol{V}_i} \tag{A5}$$

$$\overline{\mathbf{G}}_i = \sum_{j=1}^{N_{sub}} \left(\frac{\partial \mathbf{V}_{C_j}}{\partial \boldsymbol{V}_i} \right)^\top \left[\frac{\mathrm{d}}{\mathrm{d}t} \left(\mathcal{M}_j \frac{\partial \mathbf{V}_{C_j}}{\partial \boldsymbol{V}_i} \right) + \widetilde{\mathbf{\Omega}}_j \mathcal{M}_j \frac{\partial \mathbf{V}_{C_j}}{\partial \boldsymbol{V}_i} \right] \tag{A6}$$

$$\overline{\mathbf{Q}}_i = \sum_{j=1}^{N_{sub}} \left(\frac{\partial \mathbf{V}_{C_j}}{\partial \boldsymbol{V}_i} \right)^\top \mathcal{W}_j. \tag{A7}$$

Here, $\overline{\mathbf{G}}_i$ is the subsystem version of the gyroscopic matrix and $\overline{\mathbf{Q}}_i$ is the overall vector of subsystem forces. This formulation is applicable to system containing flexible bodies [14]. Introducing the relation $\boldsymbol{V}_i = \frac{\partial \boldsymbol{V}_i}{\partial \dot{\mathbf{q}}} \dot{\mathbf{q}}$ of subsystem velocities and generalized velocities, in (A7), the principle of virtual power yields the overall EoM (10) of the multibody system, where the term $\overline{\mathbf{h}}_i := \overline{\mathbf{G}}_i \boldsymbol{V}_i - \overline{\mathbf{Q}}_i$ is used throughout the paper for brevity.

Author contributions G.K. wrote the manuscript text and prepared all the necessary data and figures. H.G. and A.M. were involved in developing the formulations and in the preparations and revision of the paper.

Funding Open access funding provided by Johannes Kepler University Linz. This work has been supported by the "LCM - K2 Center for Symbiotic Mechatronics" within the framework of the Austrian COMET-K2 program.

Data Availability No datasets were generated or analysed during the current study.

Declarations

Competing interests The authors declare no competing interests.

References

1. Khatib, O.: A unified approach for motion and force control of robot manipulators: the operational space formulation. IEEE J. Robot. Autom. **3**(1), 43–53 (1987)
2. Hu, Q., Xu, L., Zhang, A.: Adaptive backstepping trajectory tracking control of robot manipulator. J. Franklin Inst. **349**(3), 1087–1105 (2012)
3. Del Prete, A.: Joint position and velocity bounds in discrete-time acceleration/torque control of robot manipulators. IEEE Robot. Autom. Lett. **3**(1), 281–288 (2018)
4. Wensing, P.M., Posa, M., Hu, Y., Escande, A., Mansard, N., Del Prete, A.: Optimization-based control for dynamic legged robots. IEEE Trans. Robot. **40**, 43–63 (2023)
5. Abe, Y., Da Silva, M., Popović, J.: Multiobjective control with frictional contacts. In: Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (2007)
6. Bouyarmane, K., Kheddar, A.: Using a multi-objective controller to synthesize simulated humanoid robot motion with changing contact configurations. In: 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (2011)
7. Koolen, T., Smith, J., Thomas, G., Bertrand, S., Carff, J., Mertins, N., Stephen, D., Abeles, P., Englsberger, J., McCrory, S., Egmond, J., Griffioen, M., Floyd, M., Kobus, S., Manor, N., Alsheikh, S., Duran, D., Bunch, L., Morphis, E., Colasanto, L., Hoang, K.-L.H., Layton, B., Neuhaus, P., Johnson, M., Pratt, J.: Summary of team IHMC's virtual robotics challenge entry. In: 2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids) (2013)
8. Vaillant, J., Kheddar, A., Audren, H., Keith, F., Brossette, S., Escande, A., Bouyarmane, K., Kaneko, K., Morisawa, M., Gergondet, P., et al.: Multi-contact vertical ladder climbing with an HRP-2 humanoid. Auton. Robots **40**(3), 561–580 (2016)
9. Wensing, P.M., Orin, D.E.: Improved computation of the humanoid centroidal dynamics and application for whole-body control. Int. J. Humanoid Robot. **13**(01), 1550039 (2016)
10. Wensing, P.M., Orin, D.E.: Generation of dynamic humanoid behaviors through task-space control with conic optimization. In: 2013 IEEE International Conference on Robotics and Automation (2013)
11. Featherstone, R.: Rigid Body Dynamics Algorithms. Springer, New York (2008)
12. Brandl, H., Johanni, R., Otter, M.: A very efficient algorithm for the simulation of robots and similar multibody systems without inversion of the mass matrix. IFAC Proc. Vol. **19**(14), 95–100 (1986)
13. Bae, D.-S., Haug, E.J.: A recursive formulation for constrained mechanical system dynamics: Part I. Open loop systems. J. Struct. Mech. **15**(3), (1987)
14. Bremer, H.: Elastic Multibody Dynamics. Springer, Dordrecht (2008)
15. Carpentier, J., Mansard, N.: Analytical derivatives of rigid body dynamics algorithms. In: Robotics: Science and Systems (RSS 2018) (2018)
16. Lilly, K.: Efficient Dynamic Simulation of Robotic Mechanisms. Springer, New York (1993)
17. Lilly, K.W., Orin, D.E.: Efficient O(N) recursive computation of the operational space inertia matrix. IEEE Trans. Syst. Man Cybern. **23**(5), 1384–1391 (1993)
18. Kreutz-Delgado, K., Jain, A., Rodriguez, G.: Recursive formulation of operational space control. Int. J. Robot. Res. **11**(4), 320–328 (1992)
19. Wensing, P., Featherstone, R., Orin, D.E.: A reduced-order recursive algorithm for the computation of the operational-space inertia matrix. In: 2012 IEEE International Conference on Robotics and Automation (2012)

20. Gattringer, H., Müller, A., Pucher, F., Reiter, A.: O(n) algorithm for elastic link/joint robots with end-effector contact. In: Zahariev, E., Cuadrado, J. (eds.) IUTAM Symposium on Intelligent Multibody Systems – Dynamics, Control, Simulation. Springer, Cham (2019)
21. Bremer, H.: Dynamik und Regelung Mechanischer Systeme. Teubner, Stuttgart (1988)
22. Gattringer, H., Oberhuber, B., Mayr, J., Bremer, H.: Recursive methods in control of flexible joint manipulators. Multibody Syst. Dyn. **32**, 117–131 (2014)
23. Siciliano, B., Khatib, O. (eds.): Springer Handbook of Robotics, vol. 200, 1st edn. pp. 55–56. Springer, Berlin (2008)