A general framework for modeling and dynamic simulation of multibody systems using factor graphs

Jose-Luis Blanco-Claraco · Antonio Leanza ·
Giulio Reina

Abstract In this paper, we present a novel general framework grounded in the factor graph theory to solve kinematic and dynamic problems for multibody systems. Although the motion of multibody systems is considered to be a well-studied problem and various methods have been proposed for its solution, a unified approach providing an intuitive interpretation is still pursued. We describe how to build factor graphs to model and simulate multibody systems using both, independent and dependent coordinates. Then, batch optimization or a fixed lag smoother can be applied to solve the underlying optimization problem that results in a highly sparse nonlinear minimization problem. The proposed framework has been tested in extensive simulations and validated against a commercial multibody software. We release a reference implementation as an open-source C++ library, based on the GTSAM framework, a well-known estimation library. Simulations of forward and inverse dynamics are presented, showing comparable accuracy with classical approaches.

The proposed factor graph-based framework has the potential to be integrated into applications related with motion estimation and parameter identification of complex mechanical systems, ranging from mechanisms to vehicles, or robot manipulators.

J-L. Blanco-Claraco (✉)
Department of Engineering, University of Almería,
CIESOL. Campus de Excelencia Internacional
Agroalimentario, ceiA3, 04120 Almería, Spain
e-mail: jlblanco@ual.es

A. Leanza
Department of Innovation Engineering, University of
Salento, via Monteroni, 73100 Lecce, Italy

G. Reina
Department of Mechanics, Mathematics, and Management,
Polytechnic of Bari, via Orabona 4, 70126 Bari, Italy

List of symbols

$\mathbf{q}(t), \dot{\mathbf{q}}(t), \ddot{\mathbf{q}}(t)$	Vector of generalized dependent coordinates, velocities, and accelerations, respectively.
$\mathbf{z}(t), \dot{\mathbf{z}}(t), \ddot{\mathbf{z}}(t)$	Vector of generalized independent coordinates, velocities, and accelerations, respectively.
$\Phi(\mathbf{q}, t)$	Vector of constraint equations.
$\Phi_{\mathbf{q}}$	Jacobian matrix of the constraints Φ.
Φ_t	Partial derivatives of constraints Φ with respect to time t.
$\Phi_{\mathbf{qq}}, (\dot{\Phi}_{\mathbf{q}})_{\mathbf{q}}$	Jacobians of $\Phi_{\mathbf{q}}$ and $\dot{\Phi}_{\mathbf{q}}$ with respect to $\mathbf{q}$ (third-order tensors).
$\mathbf{b}$	Constraint velocities.
$\mathbf{c}$	Constraint acceleration.
$\boldsymbol{\lambda}$	Vector of Lagrangian multipliers.
C	Active "branch" or "configuration" of a mechanism.

$\mathbf{Q}_t$	Generalized forces for time step t.
$\mathbf{M}$	Mass matrix for a mechanism (dependent coordinates).
$\overline{\mathbf{Q}}$	Reduced generalized forces vector.
$\overline{\mathbf{M}}$	Reduced mass matrix for a mechanism (independent coordinates).
$\mathbf{R}$	The $\mathbf{R}$ matrix, mapping increments between independent and dependent velocities.
n	Number of dependent coordinates in $\mathbf{q}(t)$.
m	Number of scalar constrain equations in Φ.
$\mathbb{R}$	The set of real numbers.

1 Introduction

Dynamic simulation of multibody systems mainly refers to inverse and forward problems. Inverse dynamics deals with the determination of the driving forces that generate a given motion, as well as the constraint reaction forces. The solution to the inverse dynamics problem can be obtained for example using the Newton–Euler (N-E) method [1] that results in efficient recursive algorithms, e.g., Rigid Body Dynamics Library (RBDL) [2] and similar methods [3]. Conversely, forward dynamics involves the motion estimation of a multibody system over time under the given applied forces and initial conditions. Therefore, in a direct dynamic problem, the motion is the result of the force system that generates it. From a mathematical perspective, forward dynamics is computationally intensive as it entails the integration of a system of nonlinear ordinary differential equations. The most common formulations that deal with forward dynamic are the Composite Rigid Body Algorithm (CRBA) [4] and the Articulated Body Algorithm (ABA) [5]. However, the above-mentioned methods are not suited for systems with closed loops like parallel robots, for which more complex and expensive procedures are required to solve both inverse and forward dynamics, as described, for example, in [6]. Therefore, a single algorithm that can solve all types of dynamics problems has not been fully established. One notable effort is the work by Rodriguez [7], who proposed a unified approach based on Kalman filtering and on the concept of smoothing to solve dynamics problems. This approach has been further extended to solve the forward dynamic problem

of closed kinematic chains [8]. Another effort in defining a unifying framework has been given by [9], who analyzed various algorithms for serial chain dynamics. In [10], an attempt is proposed to unify the CRBA and ABA derivation as two elimination methods to solve forward dynamics. To the best of our knowledge, factor graphs have not been applied to solve the general multibody kinematics and dynamics in the existing literature. The closest works are the preprints [11,12], where factor graphs are indeed employed to solve kinematic and dynamic problems, although their proposed graph structure is applicable to open-loop robot arms only.

The present work proposes factor graphs as a unifying graphical language to express kinematics and dynamics for general multibody systems. As it will be shown, classical problems such as direct and inverse dynamics can then be easily solved as especial instances of those factor graphs. Moreover, this new paradigm has the potential to develop novel and advanced state estimators. Therefore, the main contributions of this work are:

- a factor graph-based representation of dynamics problems, which is a insightful visualization of the underlying sparse relationship between all involved variables,
- a unified method, which can solve inverse and forward dynamics for either open or closed kinematic chains,
- a flexible framework that can be expressed and solved for both dependent and independent coordinates.

The implementation described in this manuscript is focused on planar mechanisms, although it is perfectly suitable to spatial systems without any change at all at the level of factor graph structures. Additionally, our approach allows more powerful and flexible schemes for state and parameter estimation to be implemented in contrast with standard methods based on Kalman filtering [13–17]. Such applications are left for future extensions of this work to keep the present manuscript focused on the key ideas on how to apply factor graphs to multibody motion problems.

The proposed approach draws on the formalism of *graphical models*, a powerful tool borrowing concepts from statistics and graph theory [18,19]. By addressing the multibody simulation problem from the perspective of the *variable structure*, graphical models allow

us creating efficient estimators for any combination of observed and hidden variables, effectively unifying the problems of kinematic and forward dynamic analysis (predicting or estimating the trajectory of a MB system), inverse dynamics, and parameter identification (e.g., inertial properties of the bodies involved, external disturbance forces, friction in the joints, etc.). All those problems end up to be formulated as a sparse nonlinear cost function built from a library of reusable "building blocks" (the *factors*) on which efficient solvers can then be applied. The framework is suitable for either offline batch analysis or online real-time operation that represents another clear advantage of the proposed approach.

The rest of the paper is structured as follows. Sections 2 and 3 first provide the required background on multibody dynamics and graphical models, respectively. Next, Sect. 4 presents a methodology for the application of Bayesian networks to multibody dynamics problems, whereas Sect. 5 particularizes such networks as factor graphs for a number of practical problems. Individual factors used in those graphs are described in detail in Sect. 6. Numerical examples are provided in Sect. 7, and we end sketching some conclusions in Sect. 8.

2 Review of multibody dynamics

In this section, fundamentals of multibody system motion analysis are briefly recalled, whereas the interested reader can refer for more details to the wide related literature, e.g., [20]. A *multibody system* is an assembly of two or more bodies (or *elements*) constrained to each other to fulfill a given motion law. In many practical applications, these elements may be considered rigid and, throughout this paper, we will work under this assumption even though the proposed framework may be further extended to include body flexibility.

One of the key decisions to take when modeling a MBS is selecting the set of *generalized coordinates* that will be used to represent it. Using *independent coordinates* $\mathbf{z}$ allows one to deal with the lowest number of parameters, i.e., the number of DOFs of the system. A second choice is to adopt *dependent coordinates* $\mathbf{q}$, in a number larger than that of DOFs but able to describe all multibody system points univocally. When dependent coordinates are used, the corresponding set of *constraint equations* must be included as well for a

complete system analysis. There are different kinds of dependent coordinates [20]: *Relative Coordinates*, *References Point Coordinates*, *Natural Coordinates*, and a combination of the previous ones (*Mixed Coordinates*). Natural coordinates, mixed with relative coordinates where needed will be assumed in this work.

In the remainder of this section, the MBS motion equations are developed and expressed in terms of both dependent and independent coordinates.

2.1 Dependent coordinates formulation

For any given MBS with f dofs, the use of n dependent coordinates expressed by the vector $\mathbf{q}$ requires $m = n - f$ constraint equations, which form the following set of equations

$$\Phi\left(\mathbf{q}(t), t\right) = \mathbf{0} \tag{1}$$

Thus, it is assumed that there are at least as many equations as there are unknown coordinates. To solve the kinematic problem, the time derivative of Eq. (1) is required, one time for velocity analysis and two times for acceleration analysis, leading to the following set of equations

$$\Phi_{\mathbf{q}}(\mathbf{q}(t), t)\,\dot{\mathbf{q}} = -\Phi_t \equiv \mathbf{b} \tag{2}$$

$$\Phi_{\mathbf{q}}(\mathbf{q}(t), t)\,\ddot{\mathbf{q}} = -\dot{\Phi}_t - \dot{\Phi}_{\mathbf{q}}\dot{\mathbf{q}} \equiv \mathbf{c} \tag{3}$$

where $\dot{\mathbf{q}}$ is the vector of dependent velocities, $\Phi_{\mathbf{q}} \in \mathbb{R}^{m \times n}$ the Jacobian matrix of Eq. (1) respect to $\mathbf{q}$, and Φ_t the time derivative of constraint equations that is equal to the null vector for time-independent constraints.

To solve the dynamic problem, external forces and inertia forces need to be considered. From the classical Newton's law, one obtains the following set of differential equations that expresses the force equilibrium equations

$$\mathbf{M}\ddot{\mathbf{q}} + \Phi_{\mathbf{q}}^{\top}\boldsymbol{\lambda} = \mathbf{Q} \tag{4}$$

where $\boldsymbol{\lambda}$ is the vector of Lagrange multipliers, $\mathbf{M}$ is the system mass matrix and vector $\mathbf{Q}$ contains the generalized external forces. Since Eq. (4) is a system of n equations in $n + m$ variables, by adding the m Eq. (3) one obtains the following system

$$\begin{bmatrix} \mathbf{M} & \Phi_{\mathbf{q}}^{\top} \\ \Phi_{\mathbf{q}} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{q}} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \mathbf{Q} \\ \mathbf{c} \end{bmatrix} \tag{5}$$

Normally [20,21], this equation is solved for the extended vector of unknowns that includes both, $\ddot{\mathbf{q}}$ and

λ. In our framework, we are specifically interested in the generalized accelerations $\ddot{\mathbf{q}}$. Therefore, by applying the block matrix inversion lemma (see 8) to Eq. (5) and keeping the first row only, it leads to the following *equation of motion* expressed in terms of *dependent coordinates*:

$$\ddot{\mathbf{q}} = \left(\mathbf{M}^{-1} - \mathbf{M}^{-1}\Phi_{\mathbf{q}}^{\top}\Gamma^{-1}\Phi_{\mathbf{q}}\mathbf{M}^{-1} \right) \mathbf{Q}$$
$$+ \left(\mathbf{M}^{-1}\Phi_{\mathbf{q}}^{\top}\Gamma^{-1} \right) \mathbf{c} \qquad (6)$$

where we defined the auxiliary term $\Gamma(\mathbf{q})$ as the $m \times m$ square symmetric matrix

$$\Gamma = \Phi_{\mathbf{q}}\mathbf{M}^{-1}\Phi_{\mathbf{q}}^{\top}, \qquad (7)$$

introduced for convenience in subsequent derivations.

2.2 Independent coordinates formulation

Independent coordinates $\mathbf{z}$, ensure the minimum number of variables, i.e., the number of DOFs. However, multibody systems can be more difficult to analyze with respect to dependent ones. First, the matrix $\mathbf{R}$ is introduced as

$$\dot{\mathbf{q}} = \frac{d\mathbf{q}}{dt} = \frac{\partial\mathbf{q}}{\partial\mathbf{z}}\frac{\partial\mathbf{z}}{\partial t} = \mathbf{R}(\mathbf{q})\dot{\mathbf{z}} \qquad (8)$$

where the columns of $\mathbf{R}$ are f linearly independent vectors that constitute a basis of the nullspace of $\Phi_{\mathbf{q}}$. Then, we express the independent velocities $\dot{\mathbf{z}}$ as the projection of the dependent velocities $\dot{\mathbf{q}}$ on the rows of a constant matrix $\mathbf{B}$ that we assume satisfying the condition of having f rows linearly independent from each other and from the m rows of $\Phi_{\mathbf{q}}$

$$\dot{\mathbf{z}} = \mathbf{B}\dot{\mathbf{q}} \qquad (9)$$

By combining Eq. (2) with Eq. (9), one obtains

$$\begin{bmatrix} \Phi_{\mathbf{q}} \\ \mathbf{B} \end{bmatrix} \dot{\mathbf{q}} = \begin{bmatrix} \mathbf{b} \\ \dot{\mathbf{z}} \end{bmatrix} \qquad (10)$$

$$\dot{\mathbf{q}} = \begin{bmatrix} \Phi_{\mathbf{q}} \\ \mathbf{B} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{b} \\ \dot{\mathbf{z}} \end{bmatrix} = \mathbf{Sb} + \mathbf{R}\dot{\mathbf{z}} \qquad (11)$$

The columns of matrix $\mathbf{R}$ are the partial velocities with respect to the generalized coordinates $\mathbf{z}$, and the term $\mathbf{Sb}$ represents the partial velocities with respect to time. By differentiating Eq. (9) and by taking into account Eq. (3), one gets

$$\begin{bmatrix} \mathbf{c} \\ \ddot{\mathbf{z}} \end{bmatrix} = \begin{bmatrix} \Phi_{\mathbf{q}} \\ \mathbf{B} \end{bmatrix} \ddot{\mathbf{q}} \Rightarrow \ddot{\mathbf{q}} = [\mathbf{S} \quad \mathbf{R}] \begin{bmatrix} \mathbf{c} \\ \ddot{\mathbf{z}} \end{bmatrix} = \mathbf{Sc} + \mathbf{R}\ddot{\mathbf{z}} \quad (12)$$

From Eq. (4), pre-multiplied by $\mathbf{R}^{\top}$

$$\mathbf{R}^{\top}\mathbf{M}\ddot{\mathbf{q}} = \mathbf{R}^{\top}\mathbf{Q} \qquad (13)$$

and by substituting $\ddot{\mathbf{q}}$ in Eq. (12)

$$\mathbf{R}^{\top}\mathbf{M}\mathbf{R}\ddot{\mathbf{z}} = \mathbf{R}^{\top}(\mathbf{Q} - \mathbf{MSc}) \qquad (14)$$

Introducing the reduced mass matrix $\overline{\mathbf{M}} = \mathbf{R}^{\top}\mathbf{MR}$ and force vector $\overline{\mathbf{Q}} = \mathbf{R}^{\top}(\mathbf{Q} - \mathbf{MSc})$, the following equation of motion in terms of independent coordinates can be finally obtained

$$\ddot{\mathbf{z}} = \overline{\mathbf{M}}^{-1}\overline{\mathbf{Q}} \qquad (15)$$

3 Background on graphical models

A factor graph is a particular type of probabilistic *graphical model* that can be used to describe the structure of an estimation problem [22]. Factor graphs have found applications in many fields, for example in robot perception [23], information theory [24], signal processing [25], and in other areas of robotics, mostly SLAM [26] and computer vision [27], state estimation in legged robots [28], and kinematic motion planning [29].

3.1 Dynamic Bayesian networks

It is instructive to start our discussion by considering a Dynamic Bayesian Network (DBN), a type of graphical model where variables are represented as nodes and *directed* edges stand for *causal* relationships [22,30]. The existence of directed edges in a DBN allows us to encode an expert's knowledge in *causality* relationships between all involved variables in the graph.

The rules to convert a DBN into the kind of graphs we are actually using in this work, factor graphs (FGs), are discussed in Sect. 5. A relevant point regarding such conversion is that one single DBN can be mapped into different FGs depending on which subset of the original DBN variables are *known data* and which *unknowns* are required to be found. Let us remark that whereas a DBN displays *all* variables as nodes, in a FG only *unknown variables* are variable nodes.

3.2 Factor graphs

Factor graphs (FGs) are bipartite graphs comprising two kinds of nodes: variable nodes and factor nodes.

Variables are the unknown data to be estimated, and factors represent any constraint (a *cost function* to be minimized) or relationship between variables. A crucial aspect of a FG is its *sparsity*: each factor node is only connected to the variables that appear in its cost function. Sparse graph optimization algorithms exist with computational cost almost linear with the number of edges in a FG, as opposed to, for example, the cubic cost of a naive implementation of Kalman Filtering [31]. Incremental (e.g., [32]) and sliding window (e.g., [33]) approaches exist as well, enabling the efficient estimation of problems with thousands to millions of variables [34–36].

Once a FG is formulated for a given problem, estimating the most-likely value of all the unknowns becomes a numerical nonlinear minimization problem, for which very efficient algorithms exist. As a probabilistic estimator, these optimal values can be assigned an uncertainty. In general, retrieving the covariance Σ of the estimated variables involves inverting the Hessian of the linearized problem evaluated at the optimal solution, such that $\Sigma = (\mathbf{J}^\top \Lambda \mathbf{J})^{-1}$ with $\mathbf{J}$ the sparse Jacobian of all constraints (factors) and Λ the weight matrix representing our level of certainty about each constraint. The matrix inversion operation is, in general, very costly since it is cubic with the problem dimension. However, optimized methods exist for the kind of sparse problem at hand, see, for example, [37, §B.4] or [32].

4 DBN for multibody dynamics

Since DBNs allow a human expert to specify the *causality* relationships between variables, we propose the two DBNs in Fig. 1 as the underlying structure of a generic multibody system that is the basis for our proposed framework. Note the use of *discrete* time steps t, exactly as simulations are commonly done following numerical integration schemes. A set of observations or measurements **o** are available from sensors installed on the system. Note, however, that the graphical model formalism is flexible enough to allow each sensor to be available at a different or even sampling rate. When using this graphical model to achieve state estimation, sensors may provide *partial* information on the system dynamics, which will be then fused with the rest of graph constraints over the system trajectory to reach the most likely values of the estimated variables. In
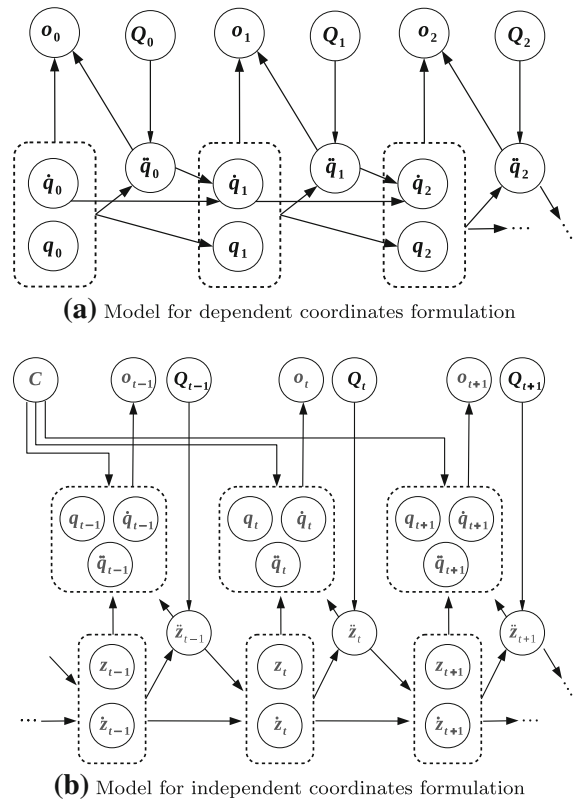

(a) Model for dependent coordinates formulation

(b) Model for independent coordinates formulation

Fig. 1 Dynamic Bayesian Network (DBN) models representing the causality relationship between the variables involved in general multibody dynamics problems over discrete time steps t. Notation: $\mathbf{q}$, $\dot{\mathbf{q}}$, $\ddot{\mathbf{q}}$ are the dependent coordinates, $\mathbf{z}$, $\dot{\mathbf{z}}$, $\ddot{\mathbf{z}}$ are independent coordinates, **o** are sensor observations, C is the active branch of the mechanism, and **Q** is the external forces. Dashed boxes represent groups of variables for which input or output directed edges affect all members. See Sect. 3 for a discussion

non-state estimation problems, the set of observations **o** may be empty. Additionally, system parameters such as masses, stiffness, or friction coefficients (constant or variable over time) could be added as additional nodes, although this possibility is left out of the scope of this manuscript for the sake of conciseness.

From Fig. 1, some observations can be drawn:

– A central role in the DBN for independent coordinates (Fig. 1b) is played by the independent coordinates $\mathbf{z}$, $\dot{\mathbf{z}}$, $\ddot{\mathbf{z}}$ as they represent the degrees of freedom (d.o.f) that govern the evolution of the MBS. This role is assumed by the dependent coordinate variables $\mathbf{q}$, $\dot{\mathbf{q}}$, $\ddot{\mathbf{q}}$ in Fig. 1a.

– The branch variable C, required uniquely in the independent-coordinate formulation, allows us to disambiguate between, e.g., the two possible con-

figurations for a four-bar linkage if we are only given the information about the crank angle. This variable could be part of the unknowns to be estimated, as already done in [38], although that work did not use the more powerful FG representation proposed here.

- Typically, nodes in a DBN are depicted as shaded or unshaded depending of whether they are "hidden" or "observed" variables, respectively, e.g., see [18,38]. The former are estimated from the latter. Since this work focuses on FGs instead, where observed variables are subsumed into *factor nodes* and unknowns become *variable nodes* [23], different FGs will be generated from the same DBNs in Fig. 1 for different multibody simulation problems; hence, it is not convenient to establish such a distinction at the DBN level yet.
- Observations $\mathbf{o}$ (sensor readings) are a function of (all, or a subset of) dependent coordinates. Typical observations that may be useful in a MBD problem are measurements obtained from gyroscopes, accelerometers, and load cells.
- External forces $\mathbf{Q}_t$ act by means of modifying accelerations $\ddot{\mathbf{q}}_t$ or $\ddot{\mathbf{z}}_t$, according to the system dynamics, expressed by Eq. (6) or Eq. (15), respectively.

Once the model of the dynamics MBS has been expressed as a graphical model, the time evolution of the system can be obtained by converting the graph into a maximum-a-posteriori (MAP) estimation problem. To this aim, one could write down the joint probability distribution $p(\phi)$ of all the involved variables $\phi = \{\mathbf{q}_{0:t}, \dot{\mathbf{q}}_{0:t}, \ddot{\mathbf{q}}_{0:t}, \mathbf{o}_{0:t}, \mathbf{Q}_{0:t}\}$ for time steps 0 to t exploiting the conditional probability encoded by the DBN (refer to, e.g., [18,22]), which for dependent coordinates (i.e., Fig. 1a) becomes:

$$p(\phi) = p(\mathbf{q}_0)p(\dot{\mathbf{q}}_0) \left(\prod_{i=0}^{t} p(\mathbf{Q}_i) \right) \left(\prod_{i=1}^{t} p(\mathbf{o}_i | \mathbf{q}_i, \dot{\mathbf{q}}_i, \ddot{\mathbf{q}}_i) \right.$$
$$\left. p(\ddot{\mathbf{q}}_i | \mathbf{q}_i, \dot{\mathbf{q}}_i, \mathbf{Q}_i) p(\mathbf{q}_i | \mathbf{q}_{i-1}, \dot{\mathbf{q}}_{i-1}) p(\dot{\mathbf{q}}_i | \ddot{\mathbf{q}}_{i-1}, \dot{\mathbf{q}}_{i-1}) \right)$$
$$(16)$$

where each conditional probability term in Eq. (16), taking negative logarithms and up to an irrelevant proportionality constant, becomes a *factor* in this alternative form of an overall *cost function* $c(\phi)$ to be minimized:

$$c(\phi) = f_{prior}(\mathbf{q}_0) f_{prior}(\dot{\mathbf{q}}_0) \left(\prod_{i=0}^{t} f_{prior}(\mathbf{Q}_i) \right)$$
$$\left(\prod_{i=1}^{t} f_{obs}(\mathbf{o}_i, \mathbf{q}_i, \dot{\mathbf{q}}_i, \ddot{\mathbf{q}}_i) f_{dyn}^{d}(\ddot{\mathbf{q}}_i, \mathbf{q}_i, \dot{\mathbf{q}}_i, \mathbf{Q}_i) \right.$$
$$\left. f_{ei}(\mathbf{q}_i, \mathbf{q}_{i-1}, \dot{\mathbf{q}}_{i-1}) f_{ei}(\dot{\mathbf{q}}_i, \ddot{\mathbf{q}}_{i-1}, \dot{\mathbf{q}}_{i-1}) \right) \quad (17)$$

A fundamental feature of graphical models is how they enable *factorizing* functions of *all* problem variables such as $c(\phi) \propto -\log p(\phi)$ into the product of a large number of *smaller* functions (in terms of the number of variables involved in each expression) called *factors* in Eq. (17), which are discussed individually in Sect. 6. This is what keeps the estimation problem tractable even for hundreds of thousands of variables, something intractable for estimators in the family of the Kalman filter not exploiting the sparsity of the problem structure. Note that the goal of an estimator is searching for the optimal set of unknowns $\phi^\star$ that maximize the likelihood of all observed data according to the model, that is:

$$\phi^\star = \arg \max_{\phi} p(\phi) = \arg \min_{\phi} c(\phi) \quad (18)$$

Depending on the division of the DBN variables into observed and unknowns, we would arrive at different factorizations since *factors* are considered to be functions of the unknowns only. For example, if all variables in one given cost factor in Eq. (17) are known data, it just evaluates to a constant which can be absorbed by the proportionality relationship $c(\phi) \propto -\log p(\phi)$ and hence can be ignored. A graphical representation of the remaining relevant factors leads us to *factor graphs* themselves.

5 Factor graphs for multibody dynamics

The conversion from DBN to FG is known to be achievable as follows, without the need to explicitly writing down probabilistic equations like Eq. (16)–Eq. (17):

Every Bayes net [DBN] node splits in both a variable node and a factor node in the corresponding factor graph. The factor is connected to the variable node, as well as the variable nodes corresponding to the parent nodes in the Bayes net. If some nodes in the Bayes net are evidence nodes, i.e., they are given as known variables, we
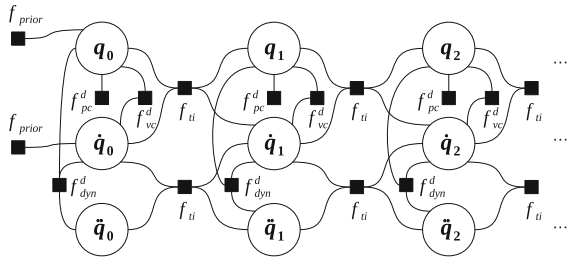
Fig. 2 Factor graphs for the forward dynamic simulation problem using dependent generalized coordinates. Circle nodes are problem unknowns; solid square nodes are factors. See discussion in Sect. 5.1

omit the corresponding variable nodes: the known variable simply becomes a fixed parameter in the corresponding factor. [23, p. 12]

Applying these rules to the DBN reflecting the structure of variables involved in the dynamic simulation of multibody systems with dependent coordinates in Fig. 1a, we derive next the FGs for a couple of practical problems depending on which are the known and unknown variables. Factors will be only briefly discussed here regarding their purpose and meaning, whereas their detailed implementations are described in Sect. 6.

5.1 FG for forward dynamics in dependent coordinates

In forward dynamics simulation, we are given a known initial state of a mechanical system (position $\mathbf{q}_0$ and velocity $\dot{\mathbf{q}}_0$), their geometric and inertial properties, and the external forces that act on it ($\mathbf{Q}_i$, $i = 0, ..., N$). For simplicity, we assume no sensors are installed in the system since this problem instance does not need them, but predicting sensor outputs could be also possible adding the corresponding nodes and factors.

The resulting FG when using dependent coordinates is shown in Fig. 2, and it involves the following *factor nodes* (or plain *factors*):

- f_{prior}: Factors imposing a given *a priori* knowledge about the attached variables, e.g., initial conditions. Prior factors can be defined for both, position and velocity.
- f_{pc}^d: Factor for position constraints in dependent coordinates. It ensures the fulfillment of mechanical holonomic constraints by keeping the state vec-

tor $\mathbf{q}$ on the manifold $\Phi(\mathbf{q}, t) = \mathbf{0}$; hence, this factor is repeated for each position node $\mathbf{q}_i$. It could be omitted for $\mathbf{q}_0$ if the enforced pose imposed by the prior factor is known to be a correct mechanism position complying with $\Phi(\cdot) = \mathbf{0}$; alternatively, the prior factor can be made to affect a minimum number of variables in $\mathbf{q}_0$ (the number of degrees of freedom), leaving f_{pc}^d in charge of recalculating the rest of the generalized coordinates. This factor, on its own, solves the so-called *position problem* [20] in multibody dynamics.

- f_{vc}^d: Factor for velocity constraints in dependent coordinates, enforcing generalized velocities $\dot{\mathbf{q}}$ to remain on the manifold $\Phi_{\mathbf{q}}\dot{\mathbf{q}} + \Phi_t = \mathbf{0}$.
- f_{dyn}^d: Factor for the dynamics equation of motion: it links external forces (known data in this FG, hence not reflected as variable nodes) with the instantaneous acceleration $\ddot{\mathbf{q}}$. It also depends on $\mathbf{q}$ and $\dot{\mathbf{q}}$ since acceleration is always a function of them too. Note in the graph how acceleration for each timestep depends on position and velocity of the same timestep only.
- f_{ti}: Trapezoidal numerical integrator factor is used twice per timestep: to integrate velocities into positions, and accelerations into velocities. Implicit integrators as the trapezoidal one are often employed in MBS simulations for their stability. However, the Euler integrator version is also devised (see Sect. 6), since explicit integrators are commonly used in real-time applications as well.

5.2 FG for forward dynamics in independent coordinates

Alternatively, one could devise a FG implementation for the forward dynamics problem when using *independent coordinates*, leading to the graph depicted in Fig. 3. In this case, the following factors are used:

- f_{prior}: In this case, they are used to impose both, an initial known dynamic state ($\mathbf{z}_0$, $\dot{\mathbf{z}}_0$) and *approximated* initial values for the full vector of dependent coordinates $\mathbf{q}_0$. The latter may be required to solve ambiguities in closed kinematic topologies, e.g., a four bar linkage, where knowing the minimum set of independent variables still leaves more than one feasible kinematic configuration.
- f_{eq}: An equality factor, used to impose a *soft* constraint between state vectors consecutive in time.
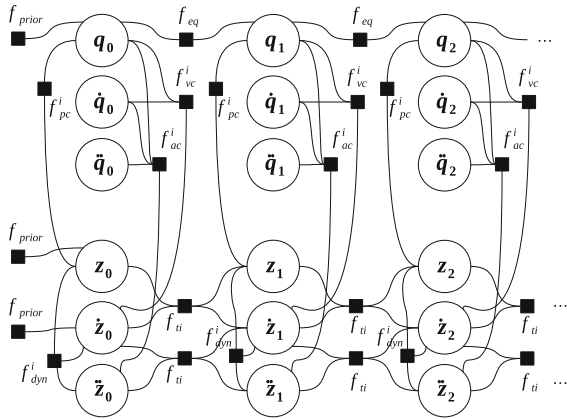
Fig. 3 Factor graph model for the forward dynamic simulation problem using independent generalized coordinates. Circle nodes are problem unknowns; solid square nodes are factors. See discussion in Sect. 5.2

Fig. 4 Factor graph for the inverse dynamics problem using dependent generalized coordinates. See discussion in Sect. 5.3

solving for the required forces and torques to generate such trajectory. The most relevant factors here are:

- f_{prior}: Different prior factors are used to define initial known values for the dynamic state ($\mathbf{q}_0$, $\dot{\mathbf{q}}_0$) and to enforce a value of zero in all components of the generalized force vectors $\mathbf{Q}_i$ where it is known that no external force is acting. In other words, the latter factors are required to leave only part of $\mathbf{Q}_i$ as an unknown, e.g., for those degrees of freedom where a motor is exerting a torque.

- f_{idyn}^d: Similar to the dynamics equation factor f_{dyn}^d discussed above, but with the force $\mathbf{Q}_i$ as an additional variable. Note that f_{dyn}^d (see Sect. 5.2) also used a *value* for $\mathbf{Q}_i$, but it was treated as a known constant, whereas for f_{dyn}^d the $\mathbf{Q}_i$ are unknowns and as such the factor provides a Jacobian of the error term with respect to them as well.

From all discussed problems so far, inverse dynamics is the hardest to solve for nonlinear iterative solvers from initial values that are far from the optimum. Therefore, it requires a proper initialization strategy to enable numerical nonlinear solvers to cope with it effectively, as discussed in the experiments section later on.

6 Factors definition

In a factor graph, factors establish the relations between variables. This section provides an insightful practical guide to those factors MBS problems are made of.

Each factor defines an *error* $\mathbf{e}(\mathbf{x})$ between predicted and measured data. To apply nonlinear optimization algorithms (e.g., Gauss–Newton or Levenberg–Marquardt), the Jacobian of such error with respect to all involved variables $\mathbf{x}$ is required. DBN variables

The rationale behind this factor is to impose a *soft constraint*, which can be easily violated (its weight is small in comparison to all other factors) but still provides a solid starting point for nonlinear numerical solvers to exploit the fact that mechanism coordinates cannot vary abruptly between consecutive time steps. This factor is particularly important to solve ambiguities in mechanisms with more than one branch, exactly as argued by the end of the former paragraph.

- f_{dyn}^i: The independent coordinate version of f_{dyn}^d discussed above, it implements the dynamics equation of motion.

- f_{pc}^i, f_{vc}^i: Like their dependent coordinate counterparts in the former section, these factors keep the position and velocity vector within their corresponding constraint manifolds. Note how, in this case, the factors not only affect $\mathbf{q}$ and $\dot{\mathbf{q}}$ coordinates, but their independent coordinate versions $\mathbf{z}$ and $\dot{\mathbf{z}}$ too, respectively. These factors actually correspond to the so-called *position problem* and *velocity problem* in multibody mechanics [20].

5.3 FG for inverse dynamics in dependent coordinates

Another problem that can be formalized as a FG is inverse dynamics, as shown in Fig. 4. The problem consists of specifying a desired trajectory over time for the set of degrees of freedom of a mechanism, then
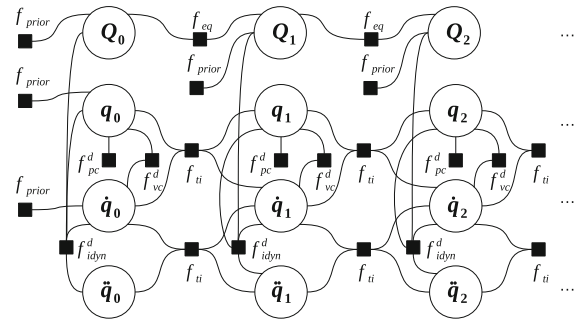
whose values are known do not become FG nodes and hence are *constant parameters* of factors. The goal of the optimization is to search for the best values of all variables $\mathbf{x}^\star$ taking the weighted sum of error functions (one per factor) as close to zero as possible:

$$\mathbf{x}^\star = \arg \min_{\mathbf{x}} \sum_i \|\mathbf{e}_i(\mathbf{x})\|_{\boldsymbol{\Sigma}}^2 \qquad (19)$$

where $\|\mathbf{e}_i(\mathbf{x})\|_{\boldsymbol{\Sigma}}^2$ is a form of *whitening* already integrating the probabilistic *noise model* (or *weight*) of each factor. The most common model, used in the present work, is the assumption of additive zero mean Gaussian noise $\mathbf{n} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$ with covariance matrix $\boldsymbol{\Sigma}$. Taking the negative logarithm of the corresponding probability density can be shown to give us the *nonlinear least-squares* problem in Eq.(19), where

$$\|\mathbf{e}(\mathbf{x})\|_{\boldsymbol{\Sigma}}^2 = \frac{1}{2}\mathbf{e}(\mathbf{x})^\top \boldsymbol{\Lambda} \mathbf{e}(\mathbf{x}) \qquad (20)$$

with information matrix $\boldsymbol{\Lambda} = \boldsymbol{\Sigma}^{-1}$. Larger information values in $\boldsymbol{\Lambda}$ (or smaller variances in $\boldsymbol{\Sigma}$) imply that the factor must be considered with a higher weight during the optimization in comparison to other factors with smaller information values (or larger variances). Note that each component of multidimensional $\mathbf{x}$ vectors may have a different weight for a given single factor, e.g., as proposed in the priors over external forces in Sect. 5.3. The interested reader is deferred to [23] for a more in-depth discussion on this topic.

In the following, note the use of the superscripts $f_{\cdot}^d$ and $f_{\cdot}^i$ for factors with differentiated implementations for dependent and independent coordinates, respectively.

6.1 f_{prior}: Prior factor

Prior factors $f_{prior}(\mathbf{x})$ over a variable $\mathbf{x}$ are the only unary factor used in our proposed graphs. They represent *a priori* belief (hence the name) about the state of a given variable.

Since the problems addressed in this manuscript use variables that are either (a) state vectors with generalized coordinates of planar mechanisms, or (b) their velocities or (c) their accelerations, all variables can be treated as vectors in the group of real numbers; hence, the error function $\mathbf{e}(\cdot)$ of this factor for a vector of dimensionality d has a simple form:

$$f_{prior} \text{ error function:} \quad \mathbf{e}(\cdot) = \mathbf{x} - \mathbf{x}_0 \qquad (21a)$$

Variables: $\quad \mathbf{x}$	(21b)
Fixed parameters: $\quad \mathbf{x}_0$	(21c)
Jacobian: $\quad \dfrac{\partial \mathbf{e}}{\partial \mathbf{x}} = \mathbf{I}_d$	(21d)

where $\mathbf{x}_0$ is the "desired" value for the variable $\mathbf{x}$.

6.2 f_{ei}: Euler integrator factor

The graphs proposed in former sections work over *temporal sequences* of variables, sampled at a fixed sample rate $f_s = 1/\Delta t$. Imposing the continuity of ordinary differential equations in discrete time can be done by means of numerical integration, of which the Euler integrator is the simplest instance.

Given two consecutive samples for time steps t and $t+1$ of a given variable $\mathbf{x}$ and its time derivative $\dot{\mathbf{x}}$ at time t, the Euler integrator factor f_{ei} (shown in Fig. 5a) can be defined as:

$$f_{ei} \text{ error function:} \quad \mathbf{e}(\cdot) = \mathbf{x}_{t+1} - \mathbf{x}_t - \Delta t \dot{\mathbf{x}}_t \quad (22a)$$

Variables: $\quad \mathbf{x}_t, \mathbf{x}_{t+1}, \dot{\mathbf{x}}_t$ (22b)

Fixed parameters: $\quad \Delta t$ (22c)

$$\text{Jacobians:} \quad \frac{\partial \mathbf{e}}{\partial \mathbf{x}_t} = -\mathbf{I}_d \quad \frac{\partial \mathbf{e}}{\partial \mathbf{x}_{t+1}} = \mathbf{I}_d$$

$$\frac{\partial \mathbf{e}}{\partial \dot{\mathbf{x}}_t} = -\Delta t \mathbf{I}_d \qquad (22d)$$

where the state space is assumed to be $\mathbb{R}^d$.

6.3 f_{ti}: Trapezoidal integrator factor

Another well-known numerical integration scheme follows the Trapezoidal Rule. Given two consecutive samples for time steps t and $t+1$, both for a given variable $\mathbf{x}$ and its time derivative $\dot{\mathbf{x}}$, the trapezoidal integrator factor f_{ti}, shown in Fig. 5b, can be defined as:

$$f_{ti} \text{ error function:} \quad \mathbf{e}(\cdot)$$

$$= \mathbf{x}_{t+1} - \mathbf{x}_t - \frac{\Delta t}{2}\dot{\mathbf{x}}_t - \frac{\Delta t}{2}\dot{\mathbf{x}}_{t+1} \quad (23a)$$

Variables: $\quad \mathbf{x}_t, \mathbf{x}_{t+1}, \dot{\mathbf{x}}_t, \dot{\mathbf{x}}_{t+1}$ (23b)

Fixed parameters: $\quad \Delta t$ (23c)

$$\text{Jacobians:} \quad \frac{\partial \mathbf{e}}{\partial \mathbf{x}_t} = -\mathbf{I}_d \quad \frac{\partial \mathbf{e}}{\partial \mathbf{x}_{t+1}} = \mathbf{I}_d$$

$$\frac{\partial \mathbf{e}}{\partial \dot{\mathbf{x}}_t} = -\frac{\Delta t}{2}\mathbf{I}_d \quad \frac{\partial \mathbf{e}}{\partial \dot{\mathbf{x}}_{t+1}} = -\frac{\Delta t}{2}\mathbf{I}_d$$

$$(23d)$$

(a) Euler integration. **(b)** Trapezoidal integration. **(c)** Position constraints **(d)** Velocity constraints
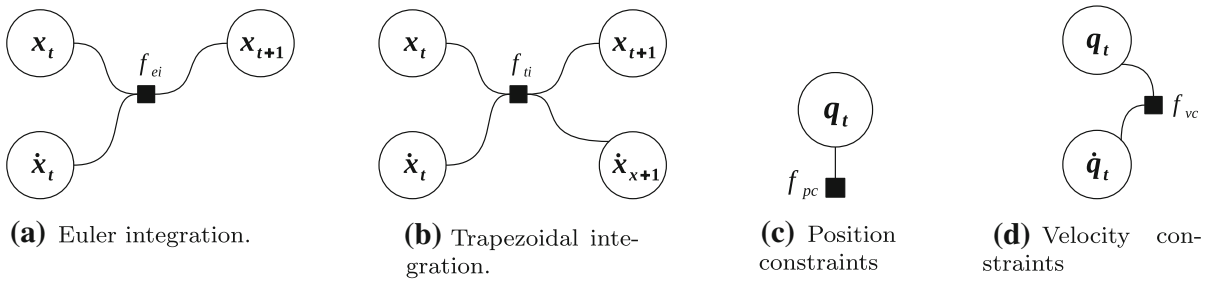
Fig. 5 **a**–**b** Numerical integration factors discussed in Sects. 6.2-6.3. Factors used to enforce $\mathbf{q}$ and $\dot{\mathbf{q}}$ to remain within their corresponding manifolds: **c** holonomic position constraints, (**b**) velocity constraints (see Sects. 6.4-6.5)

As mentioned in Sect. 5.1, this integrator is preferred for its better accuracy in comparison with the Euler method.

6.4 f_{pc}^d: Factor for position constraints in dependent coordinates

This factor, depicted in Fig. 5c, ensures the fulfillment of mechanical holonomic constraints of Eq. (1). As explained in Sect. 2.1, modeling a mechanism in dependent coordinates leads to a number of constraint equations largest or equal to the number of generalized coordinates.

This factor has the following general form:

$$f_{pc}^d \text{ error function: } \mathbf{e}(\cdot) = \Phi(\mathbf{q}_t) \quad (24a)$$
$$\text{Variables: } \mathbf{q}_t \quad (24b)$$
$$\text{Fixed parameters: Mechanism model} \quad (24c)$$
$$\text{Jacobians: } \frac{\partial \mathbf{e}}{\partial \mathbf{q}_t} = \Phi_{\mathbf{q}}(\mathbf{q}_t) \quad (24d)$$

where both Φ and $\Phi_{\mathbf{q}}$ can be automatically built out of pre-designed blocks (see Appendix III), according to a formal description of the topology of the mechanical system and the specific joints connecting each pair of adjacent bodies. In particular, since each physical constraint becomes one or more entries in the $\Phi(\mathbf{q})$ vector, each such constraint fully determines the corresponding rows in the Jacobian $\Phi_{\mathbf{q}}(\mathbf{q})$, easing its automated construction from the elemental expressions exposed in Appendix III.

6.5 f_{vc}^d: Factor for velocity constraints in dependent coordinates

Velocity constraint factors further improve the quality of MBS simulation. These kind of factors are modeled by Eq. (2) and depend on both, $\mathbf{q}$ and $\dot{\mathbf{q}}$, as illustrated in Fig. 5d. Its definition is as follows:

$$f_{vc}^d \text{ error function: } \mathbf{e}(\cdot)$$
$$= \Phi_{\mathbf{q}}(\mathbf{q}_t)\dot{\mathbf{q}}_t \underbrace{\Phi_t(\mathbf{q}_t)}_{0} = \Phi_{\mathbf{q}}(\mathbf{q}_t)\dot{\mathbf{q}}_t \quad (25a)$$
$$\text{Variables: } \mathbf{q}_t, \dot{\mathbf{q}}_t \quad (25b)$$
$$\text{Fixed parameters: Mechanism model} \quad (25c)$$
$$\text{Jacobians: } \frac{\partial \mathbf{e}}{\partial \mathbf{q}_t} = \frac{\partial \Phi_{\mathbf{q}}(\mathbf{q}_t)\dot{\mathbf{q}}_t}{\partial \mathbf{q}_t}$$
$$\frac{\partial \mathbf{e}}{\partial \dot{\mathbf{q}}_t} = \Phi_{\mathbf{q}}(\mathbf{q}_t) \quad (25d)$$

where we assumed that no constraint depends explicitly on time; hence, $\Phi_t(\mathbf{q}_t)$ can be safely neglected. Again, each row in the Jacobians comes from exactly one constraint in the definition of the mechanical system, with most common cases described in Appendix III.

6.6 f_{dyn}^d, f_{idyn}^d and f_{dyn}^i, f_{idyn}^i: Factors for dynamics

These factors minimize the error between the actual acceleration estimates ($\ddot{\mathbf{q}}$ and $\dot{\mathbf{z}}$) and the corresponding equations of motion Eqs. (6) and (15) for dependent and independent coordinates, respectively. To deal both forward and inverse dynamics, this factor connects the generalized positions, velocities, and accelerations with the forces ($\mathbf{Q}_t$) for each single time step t. In the case of forward dynamics, both the variable $\mathbf{Q}_t$ and the edge connecting it to the factor are dropped, becoming $\mathbf{Q}_t$ an internal known parameter to the factor. For this

reason, two factors for forward and inverse dynamics must be defined independently:

– **Forward dynamics**

For dependent coordinates:

f_{dyn}^d error function: $\quad \mathbf{e}(\cdot) = \ddot{\mathbf{q}}(\mathbf{q}_t, \dot{\mathbf{q}}_t) - \ddot{\mathbf{q}}_t$ (26a)

Variables: $\quad \mathbf{q}_t, \dot{\mathbf{q}}_t, \ddot{\mathbf{q}}_t$ (26b)

Fixed parameters: (masses, external forces)

(26c)

Jacobians: $\quad \dfrac{\partial \mathbf{e}}{\partial \mathbf{q}_t} = \dfrac{\partial \ddot{\mathbf{q}}(\mathbf{q}_t, \dot{\mathbf{q}}_t)}{\partial \mathbf{q}_t} \quad \dfrac{\partial \mathbf{e}}{\partial \ddot{\mathbf{q}}_t} = -\mathbf{I}$

$\dfrac{\partial \mathbf{e}}{\partial \dot{\mathbf{q}}_t} = \dfrac{\partial \ddot{\mathbf{q}}(\mathbf{q}_t, \dot{\mathbf{q}}_t)}{\partial \dot{\mathbf{q}}_t}$ (26d)

and for independent coordinates:

f_{dyn}^i error function: $\quad \mathbf{e}(\cdot) = \ddot{\mathbf{z}}(\mathbf{z}_t, \dot{\mathbf{z}}_t) - \ddot{\mathbf{z}}_t$ (27a)

Variables: $\quad \mathbf{z}_t, \dot{\mathbf{z}}_t, \ddot{\mathbf{z}}_t$ (27b)

Fixed parameters: (masses, external forces)

(27c)

Jacobians: $\quad \dfrac{\partial \mathbf{e}}{\partial \mathbf{z}_t} = \dfrac{\partial \ddot{\mathbf{z}}(\mathbf{z}_t, \dot{\mathbf{z}}_t)}{\partial \mathbf{z}_t} \quad \dfrac{\partial \mathbf{e}}{\partial \ddot{\mathbf{z}}_t} = -\mathbf{I}$

$\dfrac{\partial \mathbf{e}}{\partial \dot{\mathbf{z}}_t} = \dfrac{\partial \ddot{\mathbf{z}}(\mathbf{z}_t, \dot{\mathbf{z}}_t)}{\partial \dot{\mathbf{z}}_t}$ (27d)

– **Inverse dynamics**

For dependent coordinates:

f_{dyn}^d error function: $\quad \mathbf{e}(\cdot)$
$$= \ddot{\mathbf{q}}(\mathbf{q}_t, \dot{\mathbf{q}}_t, \mathbf{Q}_t) - \ddot{\mathbf{q}}_t \quad \text{(28a)}$$

Variables: $\quad \mathbf{q}_t, \dot{\mathbf{q}}_t, \ddot{\mathbf{q}}_t, \mathbf{Q}_t$ (28b)

Fixed parameters: (masses) (28c)

Jacobians: $\quad \dfrac{\partial \mathbf{e}}{\partial \mathbf{q}_t} = \dfrac{\partial \ddot{\mathbf{q}}(\mathbf{q}_t, \dot{\mathbf{q}}_t, \mathbf{Q}_t)}{\partial \mathbf{q}_t}$ (28d)

$\dfrac{\partial \mathbf{e}}{\partial \dot{\mathbf{q}}_t} = \dfrac{\partial \ddot{\mathbf{q}}(\mathbf{q}_t, \dot{\mathbf{q}}_t, \mathbf{Q}_t)}{\partial \dot{\mathbf{q}}_t}$ (28e)

$\dfrac{\partial \mathbf{e}}{\partial \ddot{\mathbf{q}}_t} = -\mathbf{I}$ (28f)

$\dfrac{\partial \mathbf{e}}{\partial \mathbf{Q}_t} = \dfrac{\partial \ddot{\mathbf{q}}(\mathbf{q}_t, \dot{\mathbf{q}}_t, \mathbf{Q}_t)}{\partial \mathbf{Q}_t}$ (28g)

and for independent coordinates:

f_{dyn}^i error function: $\quad \mathbf{e}(\cdot)$
$$= \ddot{\mathbf{z}}(\mathbf{z}_t, \dot{\mathbf{z}}_t, \mathbf{Q}_t) - \ddot{\mathbf{z}}_t \quad \text{(29a)}$$

Variables: $\quad \mathbf{z}_t, \dot{\mathbf{z}}_t, \ddot{\mathbf{z}}_t, \mathbf{Q}_t$ (29b)

Fixed parameters: (masses) (29c)

Jacobians: $\quad \dfrac{\partial \mathbf{e}}{\partial \mathbf{z}_t} = \dfrac{\partial \ddot{\mathbf{z}}(\mathbf{z}_t, \dot{\mathbf{z}}_t, \mathbf{Q}_t)}{\partial \mathbf{z}_t}$ (29d)

$\dfrac{\partial \mathbf{e}}{\partial \dot{\mathbf{z}}_t} = \dfrac{\partial \ddot{\mathbf{z}}(\mathbf{z}_t, \dot{\mathbf{z}}_t, \mathbf{Q}_t)}{\partial \dot{\mathbf{z}}_t}$ (29e)

$\dfrac{\partial \mathbf{e}}{\partial \ddot{\mathbf{z}}_t} = -\mathbf{I}$ (29f)

$\dfrac{\partial \mathbf{e}}{\partial \mathbf{Q}_t} = \dfrac{\partial \ddot{\mathbf{z}}(\mathbf{z}_t, \dot{\mathbf{z}}_t, \mathbf{Q}_t)}{\partial \mathbf{Q}_t}$ (29g)

Refer to Eqs. (6) and (15) for the evaluation of the error functions Eqs. (28a) and (29a), respectively. Since inverse dynamics problems seek the generalized forces in a small subset of components of the vector $\mathbf{Q}$ (i.e., those actuated with motors) given *known* values of positions $\mathbf{q}$ and velocities $\dot{\mathbf{q}}$, in our implementation we opted for implementing only the Jacobians with respect to $\mathbf{Q}_t$, that is, Eqs. (28g) and (29g). Devising closed-form expressions for these Jacobians, reported in Appendix IV, is another contribution of this work.

Not defining the other Jacobians and assuming they are zero amount to disregarding the information on $\mathbf{q}$ and $\dot{\mathbf{q}}$ *directly* inferred by the external forces $\mathbf{Q}_t$, which is a reasonable assumption in practice. Note that *indirect* information on $\mathbf{q}$ and $\dot{\mathbf{q}}$ is never lost via their time dependency on accelerations, which are indeed taken into account in this factor.

6.7 f_{pc}^i: Factor for position constraints in independent coordinates

Figure 3 shows three sets of factors linking dependent coordinates (position, velocities, and accelerations) with their corresponding degrees of freedom, that is, with their counterparts in independent coordinates. The first one of these factor is f_{pc}^i, in charge of imposing the simultaneous fulfillment of: (a) position constraints in $\Phi(\mathbf{q})$, and (b) that independent coordinates $\mathbf{z}$ are integrated into its corresponding positions within $\mathbf{q}$. Therefore, its error function and Jacobians read:

f_{pc}^i error function: $\quad \mathbf{e}(\mathbf{q}_t, \mathbf{z}_t)$
$$= \begin{bmatrix} \Phi(\mathbf{q}_t) \\ \mathbf{q}_t(\{idxs\}) - \mathbf{z}_t \end{bmatrix}_{(m+d) \times 1} \quad \text{(30a)}$$

Variables: $\quad \mathbf{q}_t, \mathbf{z}_t$ (30b)

Fixed parameters: $\{idxs\}$,

constant distances, angles, etc. (30c)

Jacobians: $\quad \dfrac{\partial \mathbf{e}}{\partial \mathbf{q}_t} = \begin{bmatrix} \Phi_{\mathbf{q}}(\mathbf{q}_t) \\ \mathbf{I}_{idxs} \end{bmatrix}_{(m+d) \times n}$

$\dfrac{\partial \mathbf{e}}{\partial \mathbf{z}_t} = \begin{bmatrix} \mathbf{0}_{m \times d} \\ -\mathbf{I}_d \end{bmatrix}_{(m+d) \times d}$ (30d)

where $\{idxs\} = \{y_1, y_2, ..., y_d\}$ stands for the fixed sequence of indices of each $\mathbf{z}$ coordinate inside the n-vector $\mathbf{q}$ and the coefficients $I_{i,j}$ of the binary matrix $\mathbf{I}_{idxs}$ are defined as 1 if $j = y_i$, or as 0 otherwise, where $i = 1, ..., d$ and $j = 1, ..., n$.

6.8 f_{vc}^i: Factor for velocity constraints in independent coordinates

This factor ensures the fulfillment of the velocity constraints for independent coordinates:

f_{vc}^i error function: $\mathbf{e}(\mathbf{q}_t, \dot{\mathbf{q}}_t, \dot{\mathbf{z}}_t)$

$$= \begin{bmatrix} \Phi_{\mathbf{q}}(\mathbf{q}_t)\dot{\mathbf{q}}_t \\ \dot{\mathbf{q}}_t(\{idxs\}) - \dot{\mathbf{z}}_t \end{bmatrix}_{(m+d)\times 1} \quad (31a)$$

Variables: $\mathbf{q}_t, \dot{\mathbf{q}}_t, \dot{\mathbf{z}}_t$ (31b)

Fixed parameters: $\{idxs\}$,

constant distances, angles, etc. (31c)

Jacobians: $\dfrac{\partial \mathbf{e}}{\partial \mathbf{q}_t} = \begin{bmatrix} \Phi_{\mathbf{qq}}(\mathbf{q}_t)\dot{\mathbf{q}}_t \\ \mathbf{0}_{d\times n} \end{bmatrix}_{(m+d)\times n}$

$\dfrac{\partial \mathbf{e}}{\partial \dot{\mathbf{q}}_t} = \begin{bmatrix} \Phi_{\mathbf{q}}(\mathbf{q}_t) \\ \mathbf{I}_{idx} \end{bmatrix}_{(m+d)\times n}$

$\dfrac{\partial \mathbf{e}}{\partial \dot{\mathbf{z}}_t} = \begin{bmatrix} \mathbf{0}_{m\times d} \\ -\mathbf{I}_d \end{bmatrix}_{(m+d)\times d}$ (31d)

where the tensor vector product $\Phi_{\mathbf{qq}}\dot{\mathbf{q}}_t$ results in the 2-D matrix $\dot{\Phi}_{\mathbf{q}}$ (see Eq. (40a)) which can be systematically built row by row from a description of the mechanism from the building blocks described in Appendix III.

6.9 f_{ac}^i: Factor for acceleration constraints in independent coordinates

This factor ensures the fulfillment of the acceleration constraints for independent coordinates:

f_{vc}^i error function: $\mathbf{e}(\mathbf{q}_t, \dot{\mathbf{q}}_t, \ddot{\mathbf{q}}_t, \ddot{\mathbf{z}}_t)$

$$= \begin{bmatrix} \dot{\Phi}_{\mathbf{q}}(\mathbf{q}_t)\dot{\mathbf{q}}_t + \Phi_{\mathbf{q}}(\mathbf{q}_t)\ddot{\mathbf{q}}_t \\ \ddot{\mathbf{q}}_t(\{idxs\}) - \ddot{\mathbf{z}}_t \end{bmatrix}_{(m+d)\times 1} \quad (32a)$$

Variables: $\mathbf{q}_t, \dot{\mathbf{q}}_t, \ddot{\mathbf{q}}_t, \ddot{\mathbf{z}}_t$ (32b)

Fixed parameters: (constant distances,

angles, etc.) (32c)

Jacobians: $\dfrac{\partial \mathbf{e}}{\partial \mathbf{q}_t} = \begin{bmatrix} (\dot{\Phi}_{\mathbf{q}})_{\mathbf{q}}(\mathbf{q}_t)\dot{\mathbf{q}}_t + \Phi_{\mathbf{qq}}(\mathbf{q}_t)\ddot{\mathbf{q}}_t \\ \mathbf{0}_{d\times n} \end{bmatrix}_{(m+d)\times n}$

$\dfrac{\partial \mathbf{e}}{\partial \dot{\mathbf{q}}_t} = \begin{bmatrix} 2\Phi_{\mathbf{qq}}(\mathbf{q}_t)\dot{\mathbf{q}}_t \\ \mathbf{0}_{d\times n} \end{bmatrix}_{(m+d)\times n}$

$\dfrac{\partial \mathbf{e}}{\partial \ddot{\mathbf{q}}_t} = \begin{bmatrix} \Phi_{\mathbf{q}}(\mathbf{q}_t) \\ \mathbf{I}_{idx} \end{bmatrix}_{(m+d)\times n}$

$\dfrac{\partial \mathbf{e}}{\partial \ddot{\mathbf{z}}_t} = \begin{bmatrix} \mathbf{0}_{m\times d} \\ -\mathbf{I}_d \end{bmatrix}_{(m+d)\times d}$ (32d)

where the tensor vector products $(\dot{\Phi}_{\mathbf{q}})_{\mathbf{q}}(\mathbf{q}_t)\dot{\mathbf{q}}_t$, $\Phi_{\mathbf{qq}}\ddot{\mathbf{q}}_t$, and $2\Phi_{\mathbf{qq}}\dot{\mathbf{q}}_t = 2\dot{\Phi}_{\mathbf{q}}$ can be systematically built as described in Appendix III.

7 Test cases validation

Firstly, a four-bar linkage is employed to exemplify the implementation and the performance of the proposed FG-based framework. The mechanism is shown in Fig. 6a, where two revolute joints, P_1 and P_2, can move in the motion plane, whereas A and D are fixed.

Geometric and inertial properties are summarized in Fig. 6b. The motion of the mechanism has been simulated using the commercial MBS environment Adams / View from *MSC.Software* with a fixed-step integrator at 1/10 of the time step used in the factor graph;

(a)
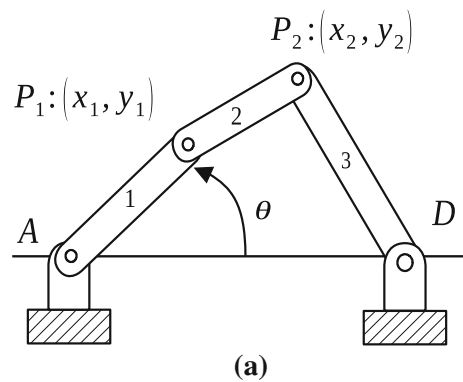
L_1	1.0	[m]
L_2	2.0	[m]
L_3	$\sqrt{13}$	[m]
m_1	1.0	[kg]
m_2	2.0	[kg]
m_3	4.0	[kg]
$I_{i,g}$	$\frac{1}{12}m_i L_i^2$	[kg m^2]

(b)

Fig. 6 **a** First planar mechanism used in the numeric simulations described in Sect. 7. **b** Table of mechanism properties (length and mass) for each link. Inertias I_i are given with respect to the center of mass of each link, placed at its center

hence, its results can be considered the ground truth for comparison purposes.

A companion open-source reference C++ implementation[1] of the proposed factor graph approach has been released along with the paper, together with tools to easily define and simulate mechanisms via human-friendly YAML files, as in the example of Listing 1.

Note that all factor closed-form Jacobians have been thoroughly validated by means of unit tests against numerical finite differences.

7.1 Forward dynamic simulation

Free motion of the four-bar mechanism under the sole effect of gravity ($g = 9.8$ m/s^2) has been simulated, with an initial crank angle θ (refer to Fig. 6a) of zero and null initial velocities.

The FG in Fig. 2 using dependent coordinates has been implemented and then solved numerically in an incremental fashion using a fixed lag smoother, when only the factors of the last N_w time steps (the *window length*) up to time t are considered, with discrete time t ranging from 0 up to 5 seconds with steps of $dt = 1$ ms. A Levenberg–Marquardt nonlinear iterative solver algorithm has been selected for these runs, with a maximum number of iterations of 15. Note, however, that virtually all calls to the optimizer required only 3 to 5 iterations until convergence.

Figure 7a, b shows the time history of the dependent generalized coordinates of points P_1 and P_2 and their corresponding velocities, respectively, obtained from both, the proposed FG-based approach and from MSC Adams. Both series accurately coincide; hence, a more detailed comparison is in order in Fig. 7c, d, where the differences in $\mathbf{q}$ and $\dot{\mathbf{q}}$ are reported between the proposed FG-based method and the ground truth. Similar good agreement (omitted for brevity's sake) is obtained when adopting the independent coordinates FG, as explained in Fig. 3, thus verifying our two proposed schemes. The noise models used in all factors involved are summarized in Table 2, whose absolute values are not relevant and have no special physical meaning, since all that matters is their relative weight with respect to each other. Note that FG optimal solutions are robust against changes in those noise models

---

```yaml
%YAML 1.2
---
# A four bars mecahnism
# Degrees of freedom in q=[x1 y1 x2 y2]^T
# Modeled in Natural coordinates:
#
#            2:(q3,q4)
#            -+----- o
#            |       (xb,yb)
#            |
#            |
#   -o----- + 1:(q1,q2)
#    (xa,ya)
#
parameters:
  L: 1.0   # length [m]
  xb: 4.0
points:
  # 0 (="A")
  - {x: 0, y: 0, fixed: true}
  # 1
  - { x: L, y: 0 }
  # 2
  - { x: L, y: 2*L }
  # 3 (="B")
  - { x: xb, y: 0, fixed: true }
planar_bodies:
  # 0
  - points: [0, 1]
    length: L
    mass: 1.0
    I0: (1/3)*mass*length^2
    cog: [0.5*length, 0.0]
  # 1
  - points: [1, 2]
    length: 2*L
    mass: 2.0
    I0: (1/3)*mass*length^2
    cog: [0.5*length, 0.0]
  # 3
  - points: [2, 3]
    length: auto
    mass: 4.0
    I0: (1/3)*mass*length^2
    cog: [0.5*length, 0.0]
```

Listing 1: YAML definition of the four-bar mechanism in Figure 6, used as an input to our reference implementation.

by at least one order of magnitude; hence, there is no need for fine tuning those parameters. The fulfillment of the constraints $|\Phi(\mathbf{q})|$ over time, illustrated in Fig. 8, is ensured by means of the addition of position constraint factors to the FG.

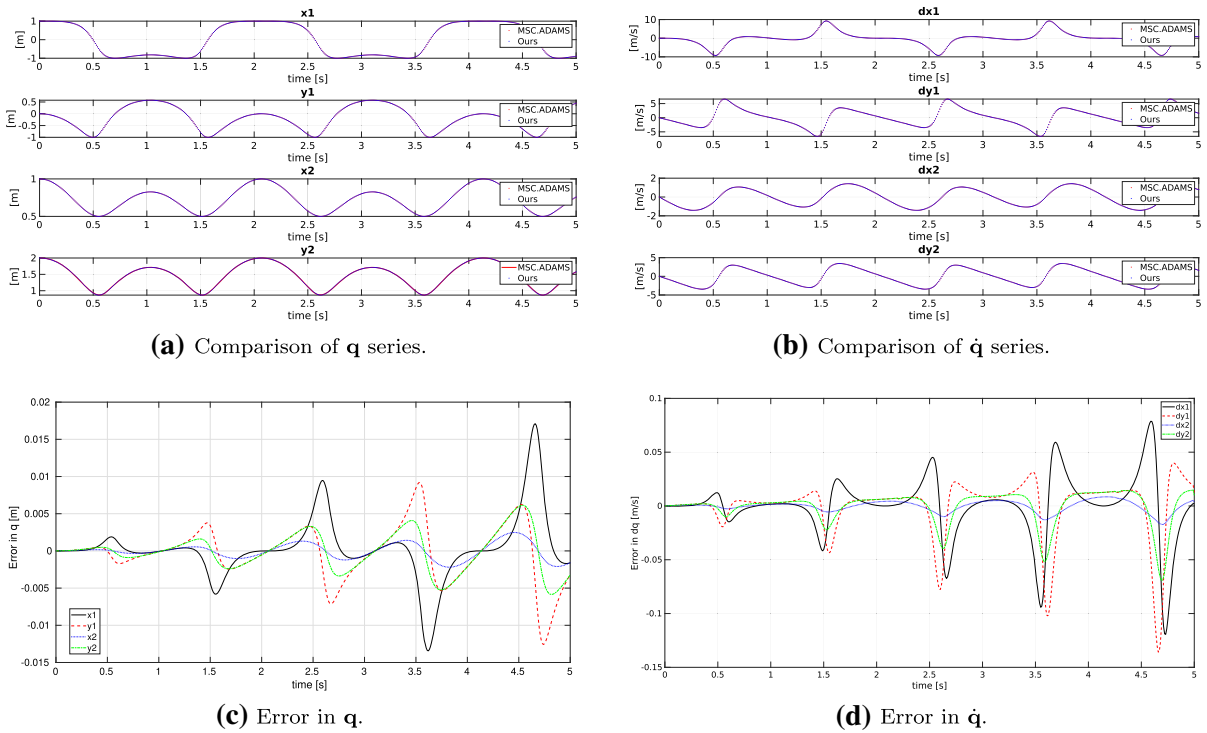In order to quantify the accuracy and computational cost of both schemes (dependent and independent-

(a) Comparison of **q** series.

(b) Comparison of **q̇** series.

(c) Error in **q**.

(d) Error in **q̇**.

Fig. 7 Simulation-based validation of the proposed FG approach to forward dynamics for a four-bar linkage under the effects of gravity. Generalized coordinates (**a**) and velocities (**b**) are shown for our approach (using a fixed-lag smoother with $N_w = 2$, time step of 1 ms) and for MSC.ADAMS as ground truth. **c** and **d** show

the error as the difference between ground truth and the output of the proposed FG-based method. Overall RMSE for this case is ≈ 0.0031 m and ≈ 0.026 m/s for **q** and **q̇**, respectively. See discussion in Sect. 7.1
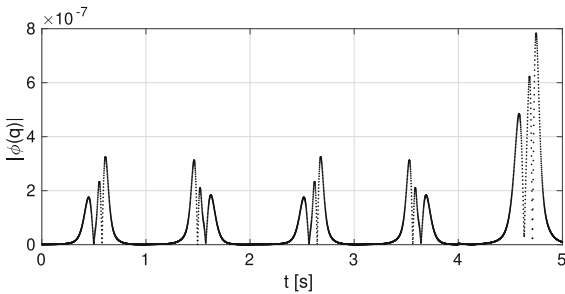
Fig. 8 Evolution over time of the L_2 norm of the constraints vector $\Phi(\mathbf{q})$ for the forward dynamics experiment for the four-bar linkage

based FGs) and to explore the importance of the fixed lag window length parameter N_w for this problem, the root mean square error (RMSE) of the overall **q** trajectories is shown in Table 1 for window sizes varying from 2 to 10 time steps, together with the average execution time for each simulation time step measured in a single thread of an Intel(R) Core(TM) i7-8700 CPU @

3.20GHz. Three conclusions can be drawn from these data: (i) both schemes, dependent and independent coordinates, seem to provide comparable accuracy, (ii) the dependent coordinates FG is more efficient (which comes at no surprise since it has less variables and factors than the independent coordinate version), and (ii) using large window lengths does not help increasing the accuracy of the estimation. This latter point can be explained by the fact that the present experiments focus on forward dynamics only, not estimation from *noisy* sensor measurements, a case where it should be expected that larger windows lead to better results. This topic, however, falls out of the scope of the present paper, and it should be studied in the future. Finally, please note a summary of the noise models employed in this experiment is shown in Table 2.

7.2 Inverse dynamics simulation

The FG proposed in Fig. 4 for inverse dynamic calculation is firstly put at test with the same four-bar

Table 1 RMSE for **q** trajectories estimated by the dependent coordinates (DC) and independent coordinates (IC) factor graphs (FG) with respect to ground truth for the simulation described in the text for different window lengths of the fixed lag smoother estimator, together with the execution time of each time step

N_w (time steps)	DC FG		IC FG	
	RMSE (mm)	Time	RMSE (mm)	Time
2	3.11773	1.4 ms	3.14719	2.8 ms
3	3.11807	1.6 ms	3.14719	3.3 ms
4	3.11818	1.8 ms	3.14719	3.7 ms
5	3.11822	2.1 ms	3.14719	4.2 ms
6	3.11823	2.3 ms	3.14719	4.7 ms
7	3.11824	2.5 ms	3.14719	5.5 ms
8	3.11824	2.8 ms	3.14719	6.1 ms
9	3.11824	3.0 ms	3.14719	6.4 ms
10	3.11824	3.2 ms	3.14719	7.2 ms

Table 2 Noise models (covariance matrices) employed in the forward dynamics simulations

Factor	Noise model
$f_{prior}(\mathbf{q}_0)$	$10^{-10}\,\mathbf{I}_n$
$f_{prior}(\mathbf{q}_0)$	$diag(\{10^{-3}$ if $q(i) \in \mathbf{z},\ 1$ otherwise$\}$
f_{ti}	$10^{-3}\,\mathbf{I}_n$
f_{dyn}^d, f_{dyn}^i	$10^{-3}\,\mathbf{I}_n$
f_{pc}^d, f_{vc}^d	$10^{-3}\,\mathbf{I}_m$
f_{pc}^i, f_{vc}^i	$10^{-3}\,\mathbf{I}_{m+d}$
f_{eq}	$10^2\,\mathbf{I}_n$

Most covariances represent isotropic models (those including the identity matrix **I**), while those having different weights for each coordinate are defined as $diag(\cdots)$, a diagonal matrix with the given diagonal values

mechanism described above. Here, the crank angle θ trajectory over time is specified as an arbitrary smooth curve, and the goal is to solve for the unknown torque at the crank (point *A*) that would be required to balance gravity and inertial forces as accurately following the prescribed path.

In this case, we use a Levenberg–Marquardt nonlinear solver on the FG in Fig. 4 in batch mode, as opposite to filtering or fixed-lag smoother mode. However, the position problem is strongly nonlinear, preventing solvers to easily find the global optimal of the whole FG in Fig. 4 for arbitrary initial values for all variables. Instead, we propose attacking the inverse dynamics FG in the following four stages (running the batch optimizer once after each stage), which ensure that the global optimum is always easy to find by the nonlinear solver:

– Stage 1: Include **q** variables, prior factors for the desired trajectory (f_{prior}), position constraint factors (f_{pc}^d), and soft equality factors (f_{eq}).
– Stage 2: Add variables $\dot{\mathbf{q}}$, $\ddot{\mathbf{q}}$, and numerical integration factors f_{ti}.
– Stage 3: Add velocity constraint factors (f_{vc}^d).
– Stage 4: Add generalized force variables **Q** and inverse dynamics factors (f_{idyn}^d).

Numerical results for this approach are shown in Fig. 9 for the four-bar mechanism, where the trajectories prescribed and actually followed by the mechanism crank are depicted in Fig. 9a for both the proposed approach and the commercial software MSC ADAMS. Both methods successfully solve the inverse dynamics problem by finding the torque sequence in Fig. 9b. In this case, both our method and MSC lead to an excellent stable following error, less than 10^{-3} degrees, as shown in Fig. 9c.

Secondly, we evaluated the inverse dynamics FG formulation against MSC.ADAMS on a more complex mechanism: a planar parallel robot featuring two degrees of freedom, based on [39]. It is controlled via two motors (unknown torques) at joints numbered as #1 and #12 in Fig. 10, and the goal is for the end-effector body to describe an "L"-like trajectory. Figure 11a shows the reference trajectory of the end-effector point #6, with a detailed view in Fig. 11b. As can be seen in
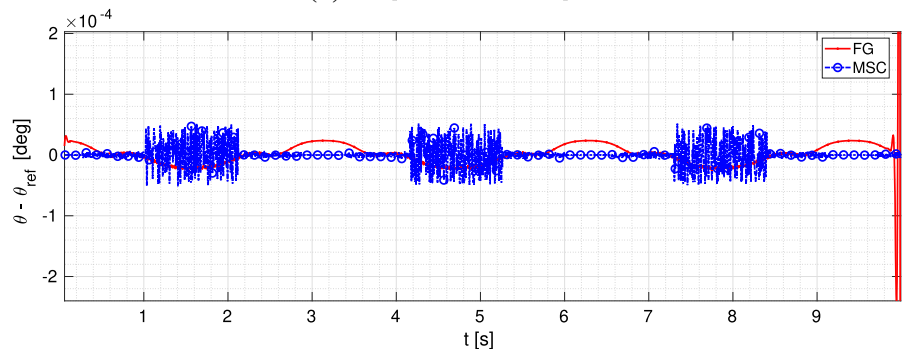
Fig. 9 Evaluation of the inverse dynamics test with the four-bar mechanism controlled with a torque at the input crank. Refer to discussion in Sect. 7.2
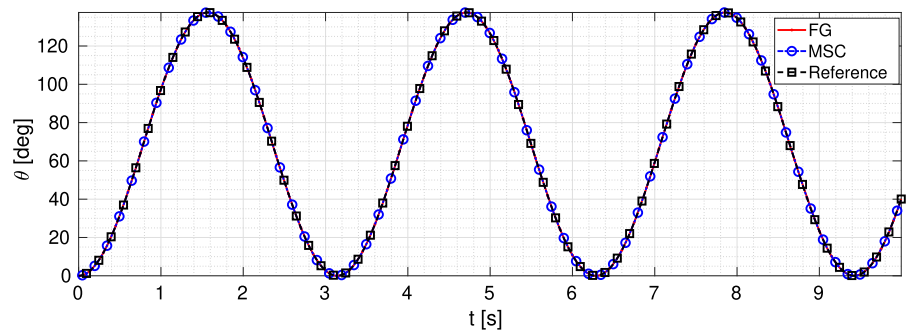
(a) Crank angle reference and simulated trajectories.
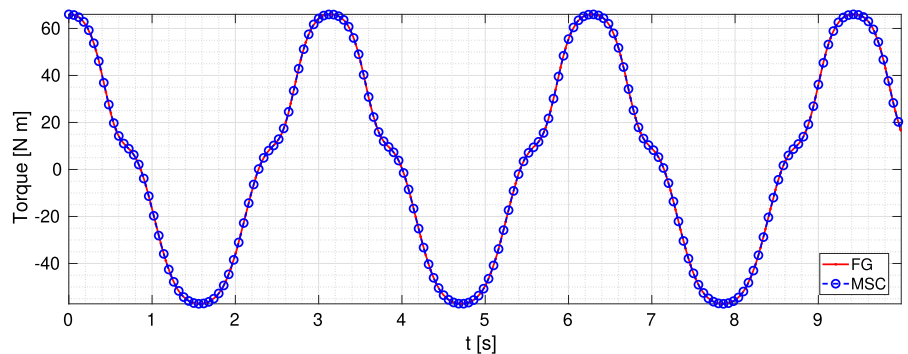
(b) Computed actuator torque.

(c) Tracking error of the crank angle.

the latter, the trajectory is accurately honored near the corner with a maximum following error of ~ 0.02 mm. Unlike the case of the four-bar mechanism above, the fact that the reference trajectory is given for a coordinate (the end-effector) that does not match where actuators are implies that inverse kinematics must be solved first to obtain the desired trajectories (angles θ_1 and θ_2) for the two motors at joints #1 and #12, respectively. The reference and the achieved trajectories for our method are represented in Fig. 11c. Regarding the inverse dynamics results themselves, the torques computed by MSC.ADAMS and our approach are plotted

in Fig. 12. It is noteworthy that our approach achieves a smoother solution, finding optimal transitions with finite jerk (i.e., acceleration derivative) in contrast with the more noisy and unrealistic acceleration obtained from MSC, which would be more problematic to transfer into an actual physical device.

A summary of relevant data to evaluate the computational cost of these inverse dynamics experiments is found in Table 3.

Table 3 Summary of performance parameters for the inverse dynamics experiments

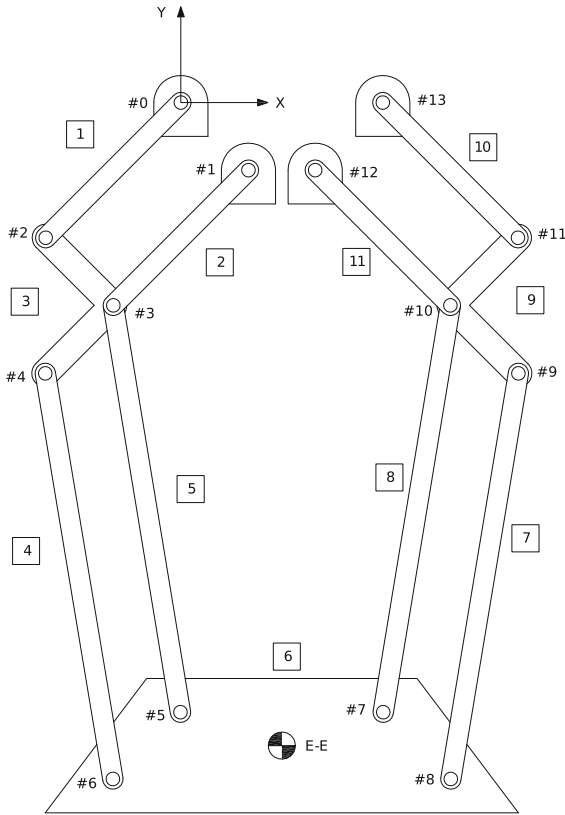| Mechanism	Generalized coordinates	Trajectory data points	CPU time
Four bar linkage	5	2000	5.3 s
Parallel robot	22	1000	15.9 s

Fig. 10 Second mechanism used as a benchmark for the proposed methods: a parallel planar robot with two degrees of freedom, modeled as 11 bodies with 14 points. The robot is actuated via two motors in points #1 and #12. Kinematic chain based on [39]

8 Conclusions and future works

This work has settled the theoretical bases for formulating kinematics and dynamics problems from computational mechanics in the form of factor graphs and demonstrated the validity of our reference open-source implementation. The results showed the practical feasibility of the proposed approach and its accuracy in comparison with a commercial MBS software based on classical approaches. One additional advantage of our formalism lies at its excellent suitability to design

state observers and parameter identification systems for arbitrary mechanical systems, ranging from robots to vehicles. This aspect will be part of future works.

Declarations

Appendix I. Block matrix inversion lemma

This lemma implies that:

$$\begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix}^{-1} = \begin{bmatrix} \mathfrak{A} & \mathfrak{B} \\ \mathfrak{C} & \mathfrak{D} \end{bmatrix} \tag{33}$$
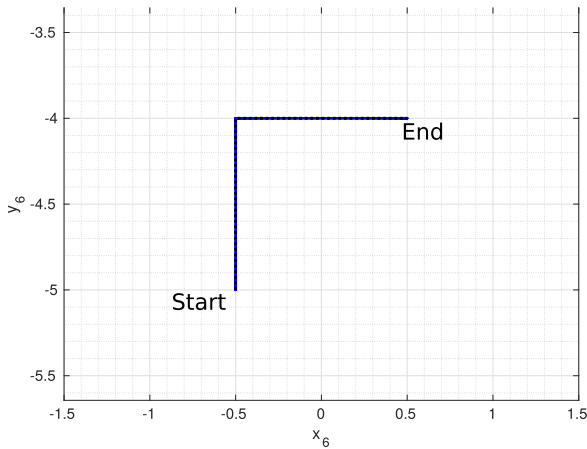
where

$$\mathfrak{A} = \mathbf{A}^{-1} + \mathbf{A}^{-1}\mathbf{B}\left(\mathbf{D} - \mathbf{C}\mathbf{A}^{-1}\mathbf{B}\right)^{-1}\mathbf{C}\mathbf{A}^{-1}$$
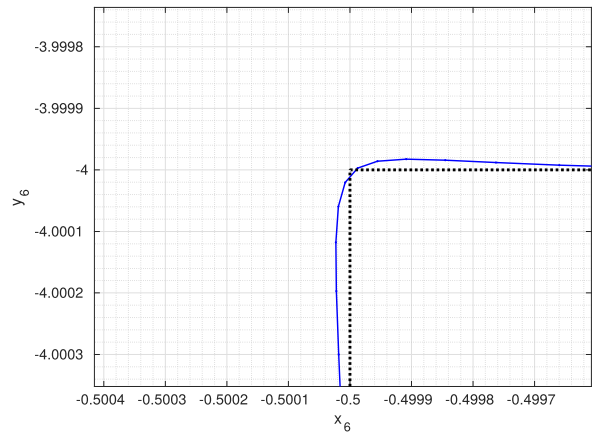
$$\mathfrak{B} = -\mathbf{A}^{-1}\mathbf{B}\left(\mathbf{D} - \mathbf{C}\mathbf{A}^{-1}\mathbf{B}\right)^{-1}$$

$$\mathfrak{C} = -\left(\mathbf{D} - \mathbf{C}\mathbf{A}^{-1}\mathbf{B}\right)^{-1}\mathbf{C}\mathbf{A}^{-1}$$
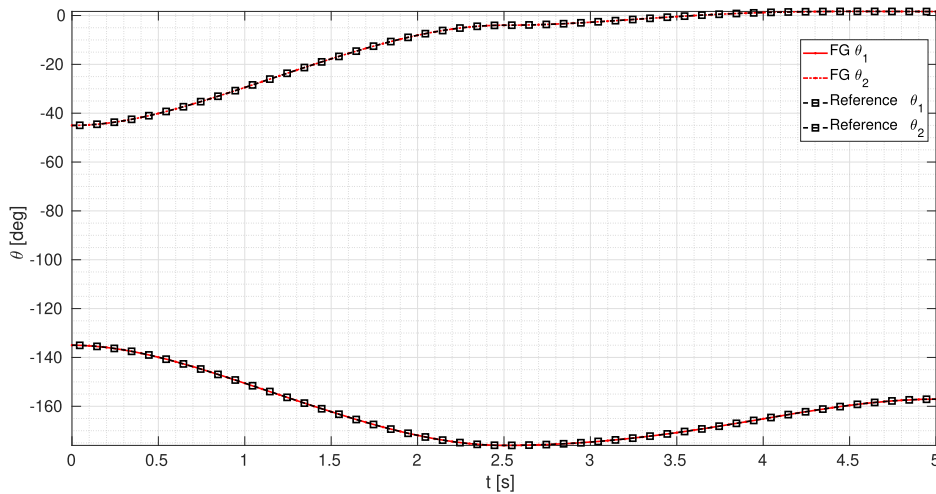
$$\mathfrak{D} = \left(\mathbf{D} - \mathbf{C}\mathbf{A}^{-1}\mathbf{B}\right)^{-1}$$

(a) Reference and actual trajectory for point "6".

(b) Detail of the trajectory near the corner.

(c) Reference and actual angles at the two motor joints.

Fig. 11 Inverse dynamics results for the pick-and-place parallel robot mechanism. The reference (black dotted) and FG-based (solid blue) trajectories for point #6 are shown in (**a**)–(**b**). The reference angles for the two motors are shown in (**c**)

Appendix II. Tensor notation definitions and useful expressions

Order of a tensor: Number of indices to index all its components, i.e., scalars, vectors, and matrices are 0th-, 1st-, and 2nd-order tensors, respectively.

Einstein convention: Superscript indices imply reading elements up–down; subscript indices imply reading left–right. Summation works over the repeated index or indices. Example with a common matrix product:

$C = AB$

$$\begin{cases} \text{Conventional notation: } C_{ik} = \sum_j A_{ij} B_{jk} \\ \text{Einstein convention: } \quad C^i{}_k = A^i{}_j B^j{}_k \end{cases} \quad (34)$$

Transpose rule: We have that $A^i{}_j = (A^\top)^j{}_i$.

Jacobian: $\frac{\partial \mathbf{x}}{\partial \mathbf{y}}$, assuming $\mathbf{x}$ and $\mathbf{y}$ are a row and a column vector, respectively, becomes:

$$\frac{\partial \mathbf{x}}{\partial \mathbf{y}} = \frac{\partial x_i}{\partial y_j} = \Delta_i{}^j \quad (35)$$

The standard derivative chain rule: Applied to vector variables $\mathbf{x}$ and $\mathbf{z}$, using an intermediary variable $\mathbf{y}$,

(a) Computed torque τ_1.

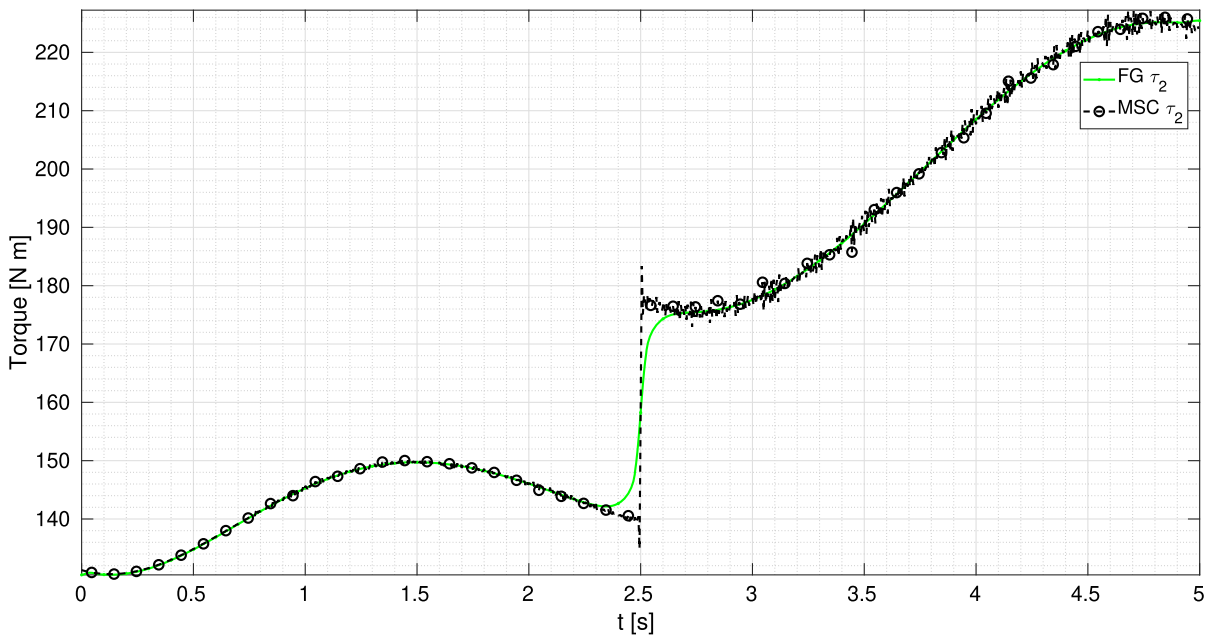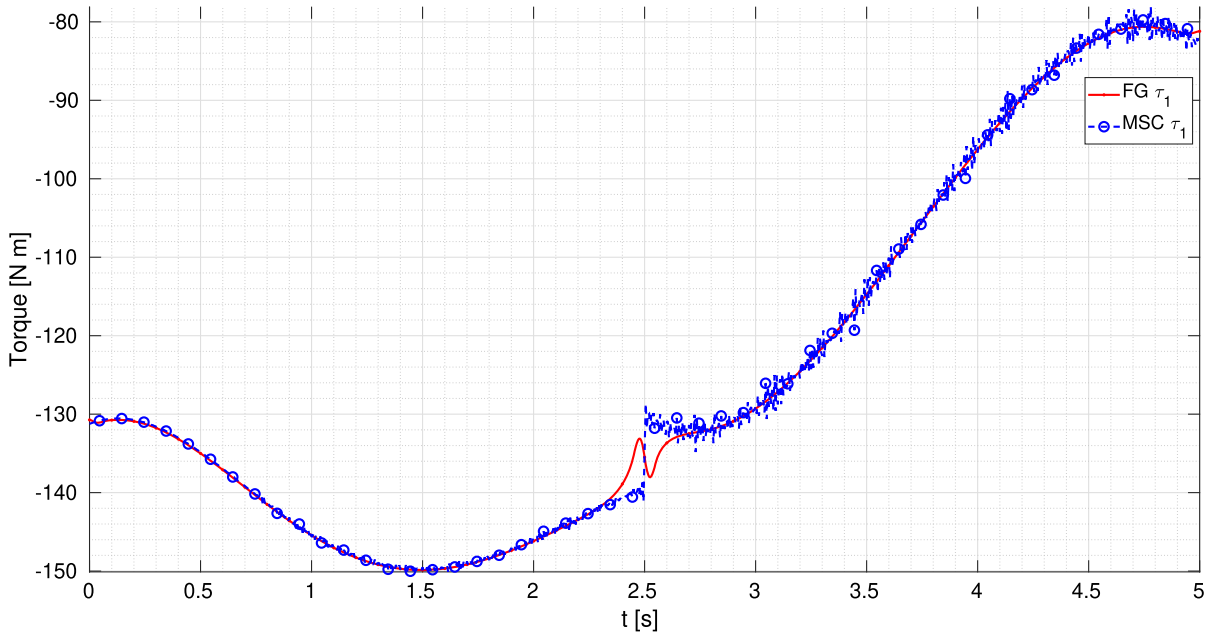**(b)** Computed torque τ_2.

Fig. 12 Inverse dynamics results for the pick-and-place parallel robot mechanism (continuation)

can be put in tensor notation as:

$$\frac{\partial \mathbf{x}}{\partial \mathbf{y}} = \frac{\partial x_i}{\partial z_k}\bigg|_{n_i \times n_k} = \frac{\partial x_i}{\partial y_j}\frac{\partial y_j}{\partial z_k} = \underbrace{\Delta^i{}_j}_{n_i \times n_j}\underbrace{\Delta^j{}_k}_{n_j \times n_k} \equiv$$

$$\equiv \sum_j \frac{\partial x_i}{\partial y_j}\frac{\partial y_j}{\partial z_k} \tag{36}$$

Chain rule for derivatives of matrices with respect to a vector: It gives us the third-order tensor:

$$\frac{\partial \mathbf{A}}{\partial \mathbf{z}} = \underbrace{\frac{\partial A^i{}_j}{\partial z_k}}_{n_i \times n_j \times n_k} = \underbrace{\frac{\partial A^i{}_j}{\partial y_l}}_{n_i \times n_j \times n_l}\underbrace{\frac{\partial y_l}{\partial z_k}}_{n_l \times n_k} \tag{37}$$

where it becomes clear that we must sum over the l indices, that is, converting the Einstein notation above into a conventional summation:

$$\frac{\partial \mathbf{A}}{\partial \mathbf{z}} = \sum_l \underbrace{\frac{\partial \mathbf{A}}{\partial y_l}}_{n_i \times n_j \times n_l}\underbrace{\frac{\partial y_l}{\partial \mathbf{z}}}_{n_l \times n_k} \tag{38}$$

Product of a third-order tensor and a vector: Given a third-order tensor $\mathbf{T} = T_j^{i\ k}$ and a vector $\mathbf{v} = v_k$, its product reads:

$$\mathbf{T}\mathbf{v} = T_j^{i\ k}\ v_k = \sum_k T_j^{ik}v_k \tag{39}$$

Appendix III. Equations for planar multibody systems

In the following, we summarize the equations implemented in our open-source library to model typical mechanical constraints of planar mechanisms. These equations are the building blocks of the errors and Jacobian matrices associated with the factors described in Sects. 6.4-6.5. An open-source implementation of all these equations is provided in the C++ toolkit mentioned in Sect. 7.

Since each constraint contributes only a few scalar components to Φ (typically only one for planar mechanisms), the following uses the superscript Φ^k to reflect the k-th row in the corresponding vector or matrix, with k being arbitrarily determined by the order in which constraints are enumerated when defining a mechanism. Furthermore, for the sake of notation clarity, $\mathbf{q}$ should be understood in the following sets of equations as the subset of the actual $\mathbf{q}$ that contains only those generalized coordinates that are directly involved in the constraint, i.e., the provided Jacobians should be expanded with columns of zeros as needed at the position of non-relevant coordinates.

In addition to the specific formulas for each constraint, the following terms appear in factor Jacobians but are generic so they apply to all constraints:

$$\dot{\Phi}_{\mathbf{q}}^k(\mathbf{q}) = \frac{d\Phi_{\mathbf{q}}^k(\mathbf{q})}{dt} = \frac{\partial \Phi_{\mathbf{q}}^k}{\partial \mathbf{q}}\dot{\mathbf{q}} + \cancel{\Phi_t}^0$$
$$= \Phi_{\mathbf{qq}}{}^k\ \dot{\mathbf{q}} \tag{40a}$$

$$\frac{\partial \dot{\Phi}_{\mathbf{q}}^k(\mathbf{q})}{\partial \dot{\mathbf{q}}} = \Phi_{\mathbf{qq}}{}^k \quad \text{(From Eq. (40a))} \tag{40b}$$

$$\frac{\partial \left(\Phi_{\mathbf{q}}^k(\mathbf{q})\,\dot{\mathbf{q}}\right)}{\partial \mathbf{q}} = \Phi_{\mathbf{qq}}{}^k\ \dot{\mathbf{q}}\ = \dot{\Phi}_{\mathbf{q}}^k(\mathbf{q}) \tag{40c}$$

$$\frac{\partial \left(\Phi_{\mathbf{q}}^k(\mathbf{q})\,\dot{\mathbf{q}}\right)}{\partial \dot{\mathbf{q}}} = \Phi_{\mathbf{q}}{}^k \tag{40d}$$

Constant distance

This constraint is typically used to model rigid solids. Given the coordinates of any pair of points i and j of a rigid body, such that $(x_i, y_i) \in \mathbf{q}$ and $(x_j, y_j) \in \mathbf{q}$, this constraint ensures a fixed distance L between them. Each such constraint contributes with the following rows in different vectors and matrices defining the multibody system:

$$\Phi^k(\mathbf{q}) = \left(x_j - x_i\right)^2 + \left(y_j - y_i\right)^2 - L^2 = 0 \tag{41a}$$

$$\Phi_{\mathbf{q}}^k(\mathbf{q}) = [2\left(x_i - x_j\right) \quad 2\left(y_i - y_j\right) \quad \cdots$$
$$\cdots \quad 2\left(x_j - x_i\right) \quad 2\left(y_j - y_i\right)] \tag{41b}$$

$$\dot{\Phi}_{\mathbf{q}}^k(\mathbf{q}) = [2\left(\dot{x}_i - \dot{x}_j\right) \quad 2\left(\dot{y}_i - \dot{y}_j\right) \quad \cdots$$
$$\cdots \quad 2\left(\dot{x}_j - \dot{x}_i\right) \quad 2\left(\dot{y}_j - \dot{y}_i\right)] \tag{41c}$$

$$\Phi_{\mathbf{qq}}{}^k(\mathbf{q}) = \frac{\partial \Phi_{\mathbf{q}}^k}{\partial \mathbf{q}}$$

$$= \left[\frac{\partial \Phi_{\mathbf{q}}^k}{\partial x_1} \quad \frac{\partial \Phi_{\mathbf{q}}^k}{\partial y_1} \quad \frac{\partial \Phi_{\mathbf{q}}^k}{\partial x_2} \quad \frac{\partial \Phi_{\mathbf{q}}^k}{\partial y_2}\right]$$

$$= \begin{matrix} & x_i & y_i & x_j & y_j & \\ \begin{bmatrix} 2 & 0 & -2 & 0 \\ 0 & 2 & 0 & -2 \\ -2 & 0 & 2 & 0 \\ 0 & -2 & 0 & 2 \end{bmatrix} & \begin{matrix} \Phi_{\mathbf{q}_1}^k \\ \Phi_{\mathbf{q}_2}^k \\ \Phi_{\mathbf{q}_3}^k \\ \Phi_{\mathbf{q}_4}^k \end{matrix} \end{matrix} \tag{41d}$$

$$(\dot{\Phi}_{\mathbf{q}})_{\mathbf{q}}^k(\mathbf{q}) = \mathbf{0}_{4\times4} \tag{41e}$$

$$\Phi_{\mathbf{qq}}{}^k(\mathbf{q})\ddot{\mathbf{q}} = [2\left(\ddot{x}_i - \ddot{x}_j\right) \quad 2\left(\ddot{y}_i - \ddot{y}_j\right) \quad \cdots$$
$$\cdots \quad 2\left(\ddot{x}_j - \ddot{x}_i\right) \quad 2\left(\ddot{y}_j - \ddot{y}_i\right)] \tag{41f}$$

Fixed pinned slider constraint

This constraints model a point $P(x, y)$ enforced to move along a line defined by two fixed points $A(x_A, y_A)$ and $B(x_B, y_B)$. By exploiting the properties of the similar triangles, one obtains:

$$\Phi^k(\mathbf{q}) = (x_B - x_A)(y - y_A) +$$
$$- (y_B - y_A)(x - x_A) \tag{42a}$$

$$\Phi_{\mathbf{q}}{}^k(\mathbf{q}) = [-(y_B - y_A) \quad (x_B - x_A)] \tag{42b}$$

$$\dot{\Phi}_{\mathbf{q}}^k(\mathbf{q}) = \mathbf{0}_{1 \times 2} \tag{42c}$$

$$\Phi_{\mathbf{qq}}{}^k(\mathbf{q}) = \mathbf{0}_{2 \times 2} \tag{42d}$$

Mobile pinned slider constraint

A more general version of the former constraint, where the point $P(x, y)$ is a now moving along a line defined by two *mobile* points $A(x_i, y_i)$ and $B(x_j, y_j)$. In this case:

$$\Phi^k(\mathbf{q}) = (x_j - x_i)(y - y_i) - (y_j - y_i)(x - x_i) \tag{43a}$$

$$\Phi_{\mathbf{q}}{}^k(\mathbf{q}) = [y_i - y_j \quad x_j - x_i \quad y_j - y \quad \cdots$$
$$\cdots \quad x - x_j \quad y - y_i \quad x_i - x] \tag{43b}$$

$$\dot{\Phi}_{\mathbf{q}}^k(\mathbf{q}) = [\dot{y}_i - \dot{y}_j \quad \dot{x}_j - \dot{x}_i \quad \dot{y}_j - \dot{y} \quad \cdots$$
$$\cdots \quad \dot{x} - \dot{x}_j \quad \dot{y} - \dot{y}_i \quad \dot{x}_i - \dot{x}] \tag{43c}$$

$$\Phi_{\mathbf{qq}}{}^k(\mathbf{q})$$

$$= \begin{bmatrix} x & y & x_i & y_i & x_j & y_j \\ 0 & 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \begin{matrix} \Phi_{\mathbf{q}_1}^k \\ \Phi_{\mathbf{q}_2}^k \\ \Phi_{\mathbf{q}_3}^k \\ \Phi_{\mathbf{q}_4}^k \\ \Phi_{\mathbf{q}_5}^k \\ \Phi_{\mathbf{q}_6}^k \end{matrix} \tag{43d}$$

$$(\dot{\Phi}_{\mathbf{q}})_{\mathbf{q}}^k(\mathbf{q}) = \mathbf{0}_{6 \times 6} \tag{43e}$$

$$\Phi_{\mathbf{qq}}{}^k(\mathbf{q})\ddot{\mathbf{q}} = [\ddot{y}_i - \ddot{y}_j \quad \ddot{x}_j - \ddot{x}_i \quad \ddot{y}_j - \ddot{y} \quad \cdots$$
$$\cdots \quad \ddot{x} - \ddot{x}_j \quad \ddot{y} - \ddot{y}_i \quad \ddot{x}_i - \ddot{x}] \tag{43f}$$

Absolute orientation of planar link

Another special kind of constraints is that one needed when defining relative coordinates, such as the absolute angle of a link, e.g., a mechanism input crank, or a vehicle wheel position. The relative angle discussed in this section is called *absolute* since it is defined as the angle θ between a body comprising a pair of points (x_i, y_i)–(x_j, y_j), at a fixed distance L, and the horizontal axis of the global frame of reference.

For this constraint, with the ordering of relevant coordinates being $\mathbf{q} = [x_i \ y_i \ x_j \ y_j \ \theta]^\top$, two sets of equations exist to avoid degeneracy. The first ones are used when $|\sin(\theta)| > 1/\sqrt{2}$:

$$\Phi^k(\mathbf{q}) = x_j - x_i - L\cos(\theta) = 0 \tag{44a}$$

$$\Phi_{\mathbf{q}}{}^k(\mathbf{q}) = [-1 \ 0 \ 1 \ 0 \ L\sin(\theta)] \tag{44b}$$

$$\dot{\Phi}_{\mathbf{q}}^k(\mathbf{q}) = [0 \ 0 \ 0 \ 0 \ L\dot{\theta}\cos(\theta)] \tag{44c}$$

$$\Phi_{\mathbf{qq}}{}^k(\mathbf{q}) = \begin{bmatrix} x_i & y_i & x_j & y_j & \theta \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & L\cos(\theta) \end{bmatrix} \begin{matrix} \Phi_{\mathbf{q}_1}^k \\ \Phi_{\mathbf{q}_2}^k \\ \Phi_{\mathbf{q}_3}^k \\ \Phi_{\mathbf{q}_4}^k \\ \Phi_{\mathbf{q}_5}^k \end{matrix} \tag{44d}$$

$$(\dot{\Phi}_{\mathbf{q}})_{\mathbf{q}}^k(\mathbf{q})$$

$$= \begin{bmatrix} x_i & y_i & x_j & y_j & \theta \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -L\dot{\theta}\sin(\theta) \end{bmatrix} \begin{matrix} \dot{\Phi}_{\mathbf{q}1}^k \\ \dot{\Phi}_{\mathbf{q}2}^k \\ \dot{\Phi}_{\mathbf{q}3}^k \\ \dot{\Phi}_{\mathbf{q}4}^k \\ \dot{\Phi}_{\mathbf{q}5}^k \end{matrix} \tag{44e}$$

$$\Phi_{\mathbf{qq}}{}^k(\mathbf{q})\ddot{\mathbf{q}} = [0 \ 0 \ 0 \ 0 \ L\ddot{\theta}\cos(\theta)] \tag{44f}$$

$$(\dot{\Phi}_{\mathbf{q}})_{\mathbf{q}}^k(\mathbf{q})\dot{\mathbf{q}} = [0 \ 0 \ 0 \ 0 \ -L\dot{\theta}^2\sin(\theta)] \tag{44g}$$

and the following alternative ones are used otherwise:

$$\Phi^k(\mathbf{q}) = y_j - y_i - L\sin(\theta) = 0 \tag{45a}$$

$$\Phi_{\mathbf{q}}{}^k(\mathbf{q}) = [0 \ -1 \ 0 \ 1 \ -L\cos(\theta)] \tag{45b}$$

$$\dot{\Phi}_{\mathbf{q}}^k(\mathbf{q}) = [0 \ 0 \ 0 \ 0 \ L\dot{\theta}\sin(\theta)] \tag{45c}$$

$$\Phi_{\mathbf{qq}}{}^k(\mathbf{q}) = \begin{bmatrix} x_i & y_i & x_j & y_j & \theta \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & L\sin(\theta) \end{bmatrix} \begin{matrix} \Phi_{\mathbf{q}_1}^k \\ \Phi_{\mathbf{q}_2}^k \\ \Phi_{\mathbf{q}_3}^k \\ \Phi_{\mathbf{q}_4}^k \\ \Phi_{\mathbf{q}_5}^k \end{matrix} \tag{45d}$$

$(\dot{\Phi}_{\mathbf{q}})_{\mathbf{q}}^k(\mathbf{q})$

$$
= \begin{bmatrix}
x_i & y_i & x_j & y_j & \theta \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & L\dot{\theta}\cos(\theta)
\end{bmatrix}
\begin{matrix}
\dot{\Phi}_{\mathbf{q}1}^k \\ \dot{\Phi}_{\mathbf{q}2}^k \\ \dot{\Phi}_{\mathbf{q}3}^k \\ \dot{\Phi}_{\mathbf{q}4}^k \\ \dot{\Phi}_{\mathbf{q}5}^k
\end{matrix}
\quad (45e)
$$

$$
\Phi_{\mathbf{qq}}{}^k(\mathbf{q})\ddot{\mathbf{q}} = \begin{bmatrix} 0 & 0 & 0 & 0 & L\ddot{\theta}\sin(\theta) \end{bmatrix} \quad (45f)
$$

$$
(\dot{\Phi}_{\mathbf{q}})_{\mathbf{q}}^k(\mathbf{q})\dot{\mathbf{q}} = \begin{bmatrix} 0 & 0 & 0 & 0 & L\dot{\theta}^2\cos(\theta) \end{bmatrix} \quad (45g)
$$

Appendix IV. Jacobians for inverse dynamics factors

For the convenience of the reader, we repeat here Eq. (6), whose Jacobians are sought:

$$
\ddot{\mathbf{q}}(\mathbf{q},\dot{\mathbf{q}},\mathbf{Q}) = \underbrace{\left(\mathbf{M}^{-1} - \mathbf{M}^{-1}\Phi_{\mathbf{q}}{}^\top\Gamma^{-1}\Phi_{\mathbf{q}}\mathbf{M}^{-1}\right)}_{\text{①}}\mathbf{Q}
$$
$$
+ \underbrace{\left(\mathbf{M}^{-1}\Phi_{\mathbf{q}}{}^\top\Gamma^{-1}\right)}_{\text{②}}\mathbf{c} \quad (46)
$$

where the dimensions of each element the subsequent derivations are: $\mathbf{q}\in\mathbb{R}^{n\times1}$, $\Phi_{\mathbf{q}}\in\mathbb{R}^{m\times n}$, $\frac{\partial\Phi_{\mathbf{q}}}{\partial\mathbf{q}}\in\mathbb{R}^{m\times n\times n}$ (third-order tensor), $\Gamma\in\mathbb{R}^{m\times m}$, $\mathbf{M}\in\mathbb{R}^{n\times n}$, and $\mathbf{Q}\in\mathbb{R}^{n\times1}$. The derivative of Eq. (46) with respect to generalized forces $\mathbf{Q}$ is straightforward and valid for both, time-variant and time-invariant constraints, since it only affects the term ①:

$$
\frac{\partial\ddot{\mathbf{q}}(\mathbf{q},\dot{\mathbf{q}},\mathbf{Q})}{\partial\mathbf{Q}} = \mathbf{M}^{-1} - \mathbf{M}^{-1}\Phi_{\mathbf{q}}{}^\top\Gamma^{-1}\Phi_{\mathbf{q}}\mathbf{M}^{-1} \quad (47)
$$

Note that this expression is only applicable to systems without relative coordinates, since they would render $\mathbf{M}$ semidefinite, invalidating its inverse. Fortunately, in practical inverse dynamics applications we normally do not need the full Jacobian with respect to *all* components in $\mathbf{Q}$, but only the subset of actuated degrees of freedom. Therefore, it is enough to use:

$$
\frac{\partial\ddot{\mathbf{q}}(\mathbf{q},\dot{\mathbf{q}},\mathbf{Q})}{\partial\mathbf{Q}} = \begin{bmatrix} \vdots \\ \mathbf{0}_{(n-d)\times n} \\ \vdots \\ \frac{\partial\ddot{\mathbf{z}}(\mathbf{z},\dot{\mathbf{z}},\mathbf{Q})}{\partial\mathbf{Q}} \end{bmatrix} \quad (48)
$$

where the rows of $\frac{\partial\ddot{\mathbf{z}}(\mathbf{z},\dot{\mathbf{z}},\mathbf{Q})}{\partial\mathbf{Q}}$, defined below, are placed at the corresponding indices of $\mathbf{Q}$ that are externally actuated.

Replacing the definitions of $\overline{\mathbf{M}}$ and $\overline{\mathbf{Q}}$ into Eq. (15) for the convenience of the reader, the dynamics equation for independent coordinates reads:

$$
\ddot{\mathbf{z}} = \overline{\mathbf{M}}^{-1}\overline{\mathbf{Q}} = \left(\mathbf{R}^\top\mathbf{M}\mathbf{R}\right)^{-1}\mathbf{R}^\top(\mathbf{Q}-\mathbf{M}\mathbf{S}\mathbf{c}) \quad (49)
$$

The Jacobian with respect to $\mathbf{Q}$ is therefore simply:

$$
\frac{\partial\ddot{\mathbf{z}}(\mathbf{z},\dot{\mathbf{z}},\mathbf{Q})}{\partial\mathbf{Q}} = \overline{\mathbf{M}}^{-1}\mathbf{R}^\top \quad (50)
$$

References

1. Siciliano, B., Sciavicco, L., Villani, L., Oriolo, G.: Robotics. Springer, Berlin (2009)
2. Raitoharju, M., Piché, R.: On computational complexity reduction methods for Kalman filter extensions. IEEE Aerosp. Electron. Syst. Mag. **34**(10), 2–19 (2019)
3. Naets, F., Pastorino, R., Cuadrado, J., Desmet, W.: Online state and input force estimation for multibody models employing extended Kalman filtering. Multibody Syst. Dyn. **32**(3), 317–336 (2014)
4. Walker, M.W., Orin, D.E.: Efficient dynamic computer simulation of robotic mechanisms. J. Dyn. Syst., Measurement Control **104**(3), 205–211 (1982). https://doi.org/10.1115/1.3139699
5. Featherstone, R., Orin, D.: Robot dynamics: equations and algorithms. In: Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065), vol. 1, pp. 826–834 vol.1 (2000). https://doi.org/10.1109/ROBOT.2000.844153. ISSN: 1050-4729
6. Wang, J., Gosselin, C.M.: A new approach for the dynamic analysis of parallel manipulators. Multibody Syst. Dyn. **2**(3), 317–334 (1998)
7. Rodriguez, G.: Kalman filtering, smoothing, and recursive robot arm forward and inverse dynamics. IEEE J. Robot. Autom. **3**(6), 624–639 (1987)
8. Rodriguez, G.: Recursive forward dynamics for multiple robot arms moving a common task object. IEEE Trans. Robot. Autom. **5**(4), 510–521 (1989)
9. Jain, A.: Unified formulation of dynamics for serial rigid multibody systems. J. Guidance, Control, Dyn. **14**(3), 531–542 (1991)
10. Ascher, U.M., Dinesh, P.K., Cloutier, B.P.: Forward dynamics, elimination methods, and formulation stiffness in robot simulation. Int. J. Robot. Res. **16**(6), 749–758 (1997)
11. Xie, M., Dellaert, F.: Batch and Incremental Kinodynamic Motion Planning using Dynamic Factor Graphs. arXiv:2005.12514 [cs] (2020)
12. Xie, M., Dellaert, F.: A Unified Method for Solving Inverse, Forward, and Hybrid Manipulator Dynamics using Factor Graphs. arXiv:1911.10065 [cs] (2020)

13. Cuadrado, J., Dopico, D., Barreiro, A., Delgado, E.: Real-time state observers based on multibody models and the extended Kalman filter. J. Mech. Sci. Technol. **23**(4), 894–900 (2009)
14. Loeliger, H.A., Dauwels, J., Hu, J., Korl, S., Ping, L., Kschischang, F.R.: The factor graph approach to model-based signal processing. Proc. IEEE **95**(6), 1295–1322 (2007)
15. Orin, D.E., McGhee, R., Vukobratovi, M., Hartoch, G.: Kinematic and kinetic analysis of open-chain linkages utilizing Newton-Euler methods. Math. Biosci. **43**(1–2), 107–130 (1979)
16. Reina, G., Paiano, M., Blanco-Claraco, J.: Vehicle parameter estimation using a model-based estimator. Mech. Syst. Signal Process. **87**(4), 227–241 (2017)
17. Sanjurjo, E., Naya, M.Á., Blanco-Claraco, J.L., Torres-Moreno, J.L., Giménez-Fernández, A.: Accuracy and efficiency comparison of various nonlinear Kalman filters applied to multibody models. Nonlinear Dyn. **88**(3), 1935–1951 (2017)
18. Bishop, C.: Pattern Recognition and Machine Learning. Springer, New York (2006)
19. Leutenegger, S., Lynen, S., Bosse, M., Siegwart, R., Furgale, P.: Keyframe-based visual-inertial odometry using nonlinear optimization. Int. J. Robot. Res. **34**(3), 314–334 (2015)
20. Forster, C., Carlone, L., Dellaert, F., Scaramuzza, D.: Imu preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation. In: Proceedings of Robotics: Science and Systems. Rome, Italy (2015). https://doi.org/10.15607/RSS.2015.XI.006
21. Shabana, A.A.: Dynamics of Multibody Systems. Cambridge University Press, Cambridge (2005)
22. Kaess, M., Johannsson, H., Roberts, R., Ila, V., Leonard, J.J., Dellaert, F.: isam2: Incremental smoothing and mapping using the Bayes tree. Int. J. Robot. Res. **31**(2), 216–235 (2012)
23. Dellaert, F., Kaess, M.: Factor graphs for robot perception. Found. Trends Robot. **6**, 1–139 (2017)
24. Worthen, A.P., Stark, W.E.: Unified design of iterative receivers using factor graphs. IEEE Trans. Inf. Theory **47**(2), 843–849 (2001)
25. Loeliger, H.A.: An introduction to factor graphs. IEEE Signal Process. Mag. **21**(1), 28–41 (2004)
26. Grisetti, G., Kümmerle, R., Stachniss, C., Burgard, W.: A tutorial on graph-based slam. IEEE Intell. Transp. Syst. Mag. **2**(4), 31–43 (2010)
27. Felis, M.L.: Rbdl: An efficient rigid-body dynamics library using recursive algorithms. Auton. Robots **41**(2), 495–511 (2017)
28. Hartley, R., Mangelson, J., Gan, L., Jadidi, M.G., Walls, J.M., Eustice, R.M., Grizzle, J.W.: Legged robot state-estimation through combined forward kinematic and preintegrated contact factors. In: 2018 IEEE International Conference on Robotics and Automation (ICRA), pp. 1–8. IEEE (2018)
29. Sugiarto, I., Conradt, J.: Discrete belief propagation network using population coding and factor graph for kinematic control of a mobile robot. In: 2013 IEEE International Conference on Computational Intelligence and Cybernetics (CYBERNETICSCOM), pp. 136–140. IEEE (2013)
30. De Jalon, J.G., Bayo, E.: Kinematic and Dynamic Simulation of Multibody Systems: the Real Time Challenge. Springer, Berlin (1994)
31. Qin, T., Li, P., Shen, S.: Vins-mono: A robust and versatile monocular visual-inertial state estimator. IEEE Trans. Robot. **34**(4), 1004–1020 (2018)
32. Jensen, F.V.: An Introduction to Bayesian Networks. UCL press, London (1996)
33. Strasdat, H., Davison, A.J., Montiel, J.M., Konolige, K.: Double window optimisation for constant time visual slam. In: 2011 International Conference on Computer Vision, pp. 2352–2359. IEEE (2011)
34. Agarwal, S., Furukawa, Y., Snavely, N., Simon, I., Curless, B., Seitz, S.M., Szeliski, R.: Building Rome in a day. Commun. ACM **54**(10), 105–112 (2011)
35. Koller, D., Friedman, N.: Probabilistic Graphical Models: Principles and Techniques. MIT press, Cambridge (2009)
36. Pastorino, R., Richiedei, D., Cuadrado, J., Trevisani, A.: State estimation using multibody models and nonlinear Kalman filters. Int. J. Non-Linear Mech. **53**, 83–90 (2013)
37. Triggs, B., McLauchlan, P.F., Hartley, R.I., Fitzgibbon, A.W.: Bundle adjustment–a modern synthesis. In: Triggs, B., Zisserman, A., Szeliski, R. (eds.) Vision Algorithms: Theory and Practice, pp. 298–372. Springer, Berlin (2000)
38. Blanco, J.L., Torres-Moreno, J.L., Giménez-Fernández, A.: Multibody dynamic systems as Bayesian networks: applications to robust state estimation of mechanisms. Multibody Syst. Dyn. **34**, 103–128 (2015)
39. Huang, T., Li, M., Li, Z., Chetwynd, D.G., Whitehouse, D.J.: Planar parallel robot mechanism with two translational degrees of freedom (2006). US Patent 7,090,458