A Unified Method for Solving Inverse, Forward, and Hybrid Manipulator Dynamics using Factor Graphs

Mandy Xie\textsuperscript{1} and Frank Dellaert\textsuperscript{2}

\textbf{Abstract}—This paper describes a unified method solving for inverse, forward, and hybrid dynamics problems for robotic manipulators with either open kinematic chains or closed kinematic loops based on factor graphs. Manipulator dynamics is considered to be a well studied problem, and various different algorithms have been developed to solve each type of dynamics problem. However, they are not easily explained in an unified and intuitive way. In this paper, we introduce factor graphs as a unifying graphical language in which not only to solve all types of dynamics problems, but also explain the classical dynamics algorithms in a unified framework.

I. INTRODUCTION & RELATED WORK

There are three main types of problems involved in the study of manipulator dynamics: inverse, forward, and hybrid problems. Inverse dynamics, which is used in control and motion planning, calculates the torques required at the joints to generate a desired trajectory of joint positions, velocities and accelerations. The Newton-Euler (N-E) [12] method is one of the key approaches in solving inverse dynamics problems since it results in very efficient recursive algorithms, such as RNEA [42, 21] and similar methods described in [49] and [57]. Forward dynamics, which is primarily used in the simulation of robotic manipulators, determines joints accelerations with torques applied at the joints, again given joint positions and velocities. Algorithms based on the inertia matrix method include the Composite-Rigid-Body Algorithm (CRBA) [60, 20] and propagation methods such as the Articulated-Body Algorithm (ABA) [18]. Finally, hybrid dynamics problems are sometimes used to incorporate prescribed motions for manipulators, and works out the unknown forces and accelerations with given forces at some joints and accelerations at other joints. Since neither inverse nor forward dynamics algorithms can be directly applied, solutions typically combine elements from both inverse and forward methods, e.g., the articulated-body hybrid dynamics algorithm described in [19].

The methods mentioned above do not apply for manipulators with kinematic loops. More sophisticated and expensive algorithms are required to solve both inverse and forward dynamics for parallel robots, which can be found in [19, 51].

There is rarely a single algorithm which can solve all three types of dynamics problems. Different algorithms have to be applied as described in [19]. Rodriguez [52, 51] built a unified framework based on the concept of filtering and smoothing to solve both inverse and forward dynamics, and such method can also be applied to closed kinematic loops dynamics as claimed in [53]. Different algorithms have to be designed and implemented, though they share a unified framework. Based on the work by Rodriguez et al. [54], Jain [33] analyzed various algorithms for serial chain dynamics in a unified formulation. Ascher et al. [4] unified the derivation of CRBA and ABA as two elimination methods which are used to solve forward dynamics.

We introduce factor graphs as a unifying graphical language in which to explain the classical dynamics algorithms, and present new algorithms derived from the graph theory underpinning sparse linear systems. Our contributions are:

- a unified method which can solve inverse, forward and hybrid dynamics for either kinematic chains or loops;
- a graph factor representation for dynamics problems, which is a insightful visualization of the underlying equations;
- the discovery of new dynamics algorithms corresponding to different elimination orderings in those graphs.

II. REVIEW OF MANIPULATOR DYNAMICS

Below we review the modern geometric view on framing robot dynamics, and follow the exposition and notation from the recent text by Lynch and Park [43]. As convincingly argued in their introduction, this geometric view pioneered by Brockett [10] and Murray et al. [48], unlocks the powerful tools of modern differential geometry to reason about robot dynamics. It will also help below in describing the various dynamics algorithms in a concise graphical representation.

Traditionally, the Newton-Euler equations of motion for a rigid body moving in space subjects to external forces and torques, can be expressed in body coordinates as

\begin{equation}
\begin{bmatrix}
  f_b \\
  \tau_b
\end{bmatrix} =
\begin{bmatrix}
  m \ddot{v}_b + \omega_b \times mv_b \\
  \mathcal{I}_b \ddot{\omega}_b + \omega_b \times \mathcal{I}_b \omega_b
\end{bmatrix}
\end{equation}

with \( m, \mathcal{I}_b, v_b, \) and \( \omega_b \) respectively the mass, inertia, linear and angular velocity expressed in body coordinate frame.

In the geometric view, we combine equations (1) and (2) to obtain an equation in terms of the six-dimensional body wrench \( \mathcal{F}_b \) and body twist \( \mathcal{V}_b \) quantities,

\begin{equation}
\mathcal{F}_b = \mathcal{G}_b \dot{\mathcal{V}}_b - [ad_{\mathcal{V}_b}]^T \mathcal{G}_b \mathcal{V}_b
\end{equation}

where the new quantities are defined as

\begin{align*}
\mathcal{V}_b &= \begin{bmatrix} \omega_b \\ v_b \end{bmatrix} \\
\mathcal{F}_b &= \begin{bmatrix} \tau_b \\ f_b \end{bmatrix} \\
\mathcal{G}_b &= \begin{bmatrix} \mathcal{I}_b & 0 \\ 0 & mI \end{bmatrix} \\
[ad_{\mathcal{V}_b}] &= \begin{bmatrix} \omega_b \\ v_b \\ \omega_b \end{bmatrix}
\end{align*}

Above \([\omega_b]\) is the skew-symmetric matrix formed from \( \omega_b \).
Closely following [43], applying this to the links of a serial manipulator and taking into account the constraints at the joints, we obtain four equations relating both link and joint quantities. In particular, the twist and acceleration \( V_i \) and \( V_i \) for the \( i \)-th link are expressed in a body-fixed coordinate frame attached to the center of mass. The wrench transmitted through joint \( i \) is denoted as \( F_i \), and \( G_i \) is the link’s inertia matrix. Without loss of generality, below we assume all rotational joints, and we then have:

\[
\begin{align*}
V_i - [Ad_{T_{i-1}(\theta_i)}]V_{i-1} - A_i \dot{\theta}_i &= 0 \quad (4) \\
\dot{V}_i - [Ad_{T_{i-1}(\theta_i)}]V_{i-1} - A_i \ddot{\theta}_i + [ad_{\theta_i}]A_i \dot{\theta}_i &= 0 \quad (5) \\
Ad_{T_{i+1}(\theta_i)}^T F_{i+1} - F_i + G_i V_i - [ad_{\theta_i}]^T G_i V_i &= 0 \quad (6) \\
F_i^T A_i - \tau_i &= 0 \quad (7)
\end{align*}
\]

where \( A_i \) is the screw axis for joint \( i \) (expressed in link \( i \)), and \( Ad_{T_{i-1}(\theta_i)} \) is the adjoint transformation associated with the transform \( T_{i-1,i} \) between the links (a function of \( \theta_i \)).

The four equations express kinodynamic constraints between link \( i \) and link \( i-1 \) imposed by joint \( i \): ((4)) describes the relationship between twist \( V_i \) and twist \( V_{i-1} \), where \( \dot{\theta}_i \) is the angular velocity of joint \( i \); ((5)) describes the constraint between acceleration \( \dot{V}_i \) of and acceleration \( \dot{V}_{i+1} \), which involves components due to joint acceleration \( \dot{\theta}_i \) and the acceleration caused by rotation; ((6)) describes the balance between the wrench \( F_i \) through joint \( i \) and the wrench \( F_{i+1} \) applied through joint \( i+1 \); ((7)) describes that the torque applied at joint \( i \) equals to the projection of wrench \( F_i \) on the screw axis \( A_i \) corresponding to joint \( i \).

III. A FACTOR GRAPH APPROACH

A factor graph [15] is a graphical model that can be used to describe the structure of sparse computational problems. It is used in constraint satisfaction [55, 23, 13], AI [50, 56, 37, 24, 36], sparse linear algebra [26, 29, 32], information theory [58, 41], combinatorial optimization [6, 7, 8], and even query theory [5, 17, 30]. A general theory specified in terms of algebraic semirings was also developed [11], and seminal work in theory proved essential computational complexity results [38] based on the existence of separator theorems for certain classes of graphs [39]. Factor graphs have been successfully applied in other areas of robotics, such as SLAM [35, 34, 15, 22], state estimation in humanoids [31], and (kinematic) motion planning [16, 47, 44].

A key idea is that we can use a factor graph to represent the structure of the dynamics constraints (4)-(7) for a particular manipulator configuration. A factor graph consists of factors and nodes, where factors correspond to the dynamics constraints, and the nodes represent the variables in each equation. Factors are only connected to the variable nodes that are featured in the corresponding dynamics constraint, revealing the sparsity structure of the system of dynamics equations. Figure 1b illustrates this for the classical Puma 560 robot shown in Figure 1a. The repetitive sparse structure of the 6R robot can be clearly observed.

The dynamics factor graph corresponding to all variables and constraints can be simplified and used to solve the different types of dynamics problems, i.e., inverse, forward, and hybrid problems. We show this in detail in the three following sections. However, due to space constraints, we use a three link RRR example for the remainder of this paper.

Also, in all three problems, we typically assume that the kinematic quantities \( q_i \) and \( \dot{\theta}_i \) are known for all joints. This in turn allows us to solve for the twist variables \( V_i \) in advance. In addition, we typically also assume that \( V_0 \) and the end-effector wrench \( F_f \) are given, as well. In the language of graphical models it is common to denote known variables as square nodes. This is illustrated in Fig. 2 for an RRR robot, which will be the starting point for the sections below.

A. Inverse Dynamics

In inverse dynamics, we are seeking the required joint torques \( \tau_i \) to realize the desired joint accelerations \( \ddot{\theta}_i \). We can construct a simplified, less cluttered, inverse dynamics graph by simply omitting all known nodes, although they remain as parameters in the factors they were connected to.

Fig. 1: (a) The Puma 560 dynamics factor graph, where black dots represent factors, and circles represent variables. (b) The PUMA 560 robot [3].
Eliminate all the \( F \) and \( \tau \) in the order \( \tau_3, \tau_1, \tau_2 \). This results in a dependence of \( F \) inverse dynamics factor graph in Fig. 3a is thereby converted to the DAG as shown in Fig. 4d.

**Algorithm 1: Inverse dynamics as shown in Fig. 4d.**

\[
\begin{align*}
1 \quad & \dot{V}_1 = [Ad_{F_1,0}(\theta_1)]V_0 + [ad_{V_1}A_1\dot{\theta}_1 + A_1\ddot{\theta}_1; \\
2 \quad & \dot{V}_2 = [Ad_{F_2,1}(\theta_2)]\dot{V}_1 + [ad_{V_2}A_2\dot{\theta}_2 + A_2\ddot{\theta}_2; \\
3 \quad & \dot{V}_3 = [Ad_{F_3,2}(\theta_3)]\dot{V}_2 + [ad_{V_3}A_3\dot{\theta}_3 + A_3\ddot{\theta}_3; \\
4 \quad & \dot{F}_3 = Ad_{F_1,3}(\theta_4)F_1 + G_3\dot{V}_3 - [ad_{V_3}^T G_3 V_3; \\
5 \quad & \dot{F}_2 = Ad_{F_3,2}(\theta_4)F_3 + G_2\dot{V}_2 - [ad_{V_2}^T G_2 V_2; \\
6 \quad & \dot{F}_1 = Ad_{F_2,1}(\theta_4)\dot{F}_2 + G_1\dot{V}_1 - [ad_{V_1}^T G_1 V_1; \\
7 \quad & \tau_1 = F_1^T A_1; \\
8 \quad & \tau_2 = F_2^T A_2; \\
9 \quad & \tau_3 = F_3^T A_3;
\end{align*}
\]

2) Solving Symbolically: Elimination can be done numerically or symbolically. A symbolic elimination step can be very simple if only one equation is involved, or rather complicated if many equations are involved. Hence, if it matters which variables are eliminated first. For example, eliminating \( \tau_3 \) above is simply a matter of rewriting (7)

\[
\dot{F}_3^T A_3 - \tau_3 = 0
\]

as

\[
\tau_3 = \dot{F}_3^T A_3
\]

However, if one were to eliminate \( F_2 \) in Fig. 2 first, it would involve three constraints (the number of factors attached to \( F_2 \)) and five variables, leading to two new constraints in those variables. That complexity will propagate to the rest of the graph, i.e., creating (symbolic) fill-in.

After elimination, back-substitution in reverse elimination order solves for the values of all intermediate quantities and the desired torques. For the example ordering, this corresponding to the chosen order above first computes the 6-dimensional twists accelerations \( \dot{V}_i \), link wrenches \( F_i \), and then the scalar torques. The back-substitution sequence corresponding to Fig. 4d can be written down as an algorithm.
The resulting algorithm for the 3R case and the chosen ordering is shown above as Algorithm 1.

The above exactly matches the forward-backward path used by the recursive Newton-Euler algorithm (RNEA) [42]. The resulting DAG can be viewed as a graphical representation of RNEA, where the green and blue color respectively correspond to the forward path and the backward path.

| Elimination Method | Average Time (µs) |
|--------------------|------------------|
| RNEA               | 26.6             |
| RNEA in RBDL       | 20.2             |
| COLAMD             | 11.0             |
| ND                 | 11.7             |

3) Solving Numerically: However, we can also construct these factor graphs on the fly, for arbitrary robot configurations, and solve them numerically. The symbolic elimination leads to very fast hard-coded dynamics algorithms, but have to be re-derived for every configuration. The numerical approach is exactly what underlies sparse linear algebra solvers, and can be extended to deal with over-constrained least-squares problems, in which the elimination algorithm corresponds to QR factorization.

For an arbitrary configuration, the numerical elimination (in arbitrary order) corresponds to a blocked Gaussian elimination where most blocks are 6 × 6, except where the (scalar) torques $\tau_i$ are concerned. For example, Fig 3b shows the sparse block-matrix corresponding to the simplified 3R inverse dynamics graph in Fig. 2. Every row in that matrix is associated with a factor in the graph, and every column with a variable node. After elimination, back-substitution corresponding to the chosen order above first computes the 6-dimensional twists accelerations $\dot{V}_i$, link wrenches $\mathcal{F}_i$, and then the scalar torques.

As already hinted at above, the cost of elimination on a factor graph can vary dramatically for different variable orderings, since different orderings lead to different DAG topologies. The amount of fill-in in turn affects the computational complexity of the elimination and back-substitution algorithms [15]. Unfortunately, finding an optimal ordering is NP-complete and already intractable for a 6R robot, so ordering heuristics are used. Table I shows that the RNEA ordering as discussed in Lynch & Park is apparently outperformed by the custom implementation in RBDL [21]. This is in turn outperformed by COLAMD [2] and nested dissection (ND) [25], which are two state of the art sparse linear algebra ordering heuristics. The reported results were obtained using GTSAM [14], a state of the art factor graph solver used extensively in the robotics community.

4) The Space of all Inverse Dynamics Algorithms: Given all of the above, it is clear that the underlying graph theory formalizes the existence of an entire space of possible inverse dynamics algorithms: for every of the (intractably many) possible variable orderings, we have both a numerical and a symbolic variant. In theory, given enough time, we can exhaustively search all orderings for a given configuration and the generate a hard-coded algorithm that is optimal for that configuration.

B. Forward Dynamics

In traditional expositions the forward dynamics problem cannot be solved as neatly as the relatively easy inverse dynamics problem. In the forward case we are seeking the joint accelerations $\ddot{\theta}$ when given $\theta$, $\dot{\theta}$, and $\tau$. Similar to the inverse dynamics factor graph, we can simplify forward dynamics factor graph as shown in Fig. 5a.

1) Solving Symbolically: In very much the same spirit as our work, Ascher et al. [4] showed that two of the most widely used forward algorithms, CRBA [20] and ABA [18], can be explained as two different elimination methods to solve the same linear system.

In our framework, CRBA and ABA can additionally be visualized as two different DAGs resulting from solving forward dynamics factor graph with two different elimination orders shown in Fig. 6a and Fig. 6b. CRBA can be explained as first eliminating all the wrenches $\mathcal{F}_i$, then eliminating all the accelerations $\dot{V}_i$, and lastly eliminating all the angular accelerations $\ddot{\theta}_i$. In contrast, in ABA we alternate eliminating the wrenches $\mathcal{F}_i$, accelerations $\dot{V}_i$ and angular accelerations $\ddot{\theta}_i$ for $i \in n \ldots 1$. The resulting DAGs can be viewed as graphical representations of CRBA and ABA, and for a given robot configuration, a custom back-substitution program can be written out in reverse elimination order.

The forward dynamics problem is usually more complicated than the inverse dynamics problem, and hence there
are more edges in the corresponding DAGs. However, better elimination orders lead to better algorithms. For example, from Figures 6a and 6b we can see that there are more edges in the CRBA DAG than in the ABA DAG, which means more computation is required using CRBA. This was already remarked upon by Ascher et al. [4], but in the graphical framework we can tell this directly by observing the DAG.

### Table II: Numerical forward dynamics for a PUMA 560

| Elimination Method | Average Time (\(\mu s\)) |
|--------------------|--------------------------|
| CRBA               | 51.8                     |
| ABA                | 25.5                     |
| COLAMD             | 11.2                     |
| ND                 | 12.1                     |

2) **Solving Numerically**: Forward dynamics for arbitrary robot configurations can be solved by constructing the factor graphs on the fly and solving them numerically, using a sparse solver such as GTSAM [14]. Similarly to the inverse case, we can use different variable ordering heuristics to explore the computational complexity of each scheme. In Table II we report on four different orderings, applied to the PUMA 650 configuration, and we can see that CRBA indeed outperforms ABA in this case. However, the two sparse linear algebra ordering heuristics COLAMD and ND outperform both, by yet another factor of two or more.

### C. Hybrid Dynamics

In hybrid dynamics problems, either \(\dot{\theta}_i\) or \(\tau_i(t)\) at each joint are given, and the task is to obtain the unknown accelerations and torques. To solve this problem, Featherstone introduced a hybrid dynamics algorithm in [19], where the set of joints for with the torques given but accelerations are unknown is denoted as “forward-dynamics joints”, and the others are denoted as “inverse-dynamics joints”.

![Fig. 7: Hybrid dynamics factor graph for a 3R robot.](image)

1) **Featherstone’s method**: Solving the hybrid dynamics using Featherstone’s algorithm can be illustrated with factor graphs using a simple 3R example. Fig. 7 shows a factor graph for the case when \(\tau_1\) is unknown while \(\dot{\theta}_1\) is given, and additionally \(\dot{\theta}_2\) and \(\dot{\theta}_3\) are unknown while \(\tau_2\) and \(\tau_3\) are given. For this combination of given and unknown values, the hybrid dynamics factor graph can be simplified to Fig. 8a.

- **Inverse Dynamics (Zero Acceleration Torques)**: Set \(\dot{\theta}_i\) as known variables, where the values are the desired accelerations for \(i = 1\), and the values are zeros for \(i = 2 \text{ and } 3\); Set \(\tau_1\) as known variables for \(i = 2 \text{ and } 3\), where the values are given; Calculate \(\tau_1\) with the inverse dynamics factor graph.

- **Forward Dynamics**: Set \(\tau_i\) as known, where the values are from zero acceleration torques for \(i = 1\), and the values for \(i = 2 \text{ and } 3\) are as given; Calculate \(\dot{\theta}_i\) for \(i = 2 \text{ and } 3\) with the forward dynamics factor graph.

- **Inverse Dynamics**: Set \(\dot{\theta}_i\) to be known variables, where the values are as given for \(i = 1\), and the values are from the last step for \(i = 2 \text{ to } 3\); Calculate \(\tau_1\) with the inverse dynamics factor graph.

### Table III: Elimination Orders for Hybrid Dynamics

| Elimination Method | Elimination Order                  |
|--------------------|------------------------------------|
| MD                 | \(\tau_1,\tau_2,\tau_3,\dot{\theta}_1,\dot{\theta}_2,\dot{\theta}_3\) |
| CUSTOM             | \(\tau_1,\tau_2,\tau_3,\dot{\theta}_1,\dot{\theta}_2,\dot{\theta}_3\) |

![Fig. 8: (a) Simplified hybrid dynamics graph and (b) corresponding block-sparse.](image)

![Fig. 9: DAGs from solving hybrid dynamics factor graph.](image)

2) **Using elimination in a Factor Graph**: It is not necessary to solve inverse and forward dynamics multiple times, because both forward-dynamics joints and inverse-dynamics joints can be solved simultaneously with the hybrid dynamics factor graph in Fig. 7. Using an elimination variable ordering generated by a minimum degree (MD) heuristic shown in Tab. III, the resulting DAG obtained is shown in Fig. 9a. For good measure, we also eliminated the factor graph with another, manually created elimination order (shown as "CUSTOM" in Tab. III), and show the corresponding DAG in Fig. 9b. The two DAGs are slightly different, but both have the same number of directed edges and hence are expected to have the same computational complexity.

Especially for hybrid problems like this, being sophisticated about variable ordering and the possible resulting paral-
lelism could yield large dividends. An important step forward in the understanding and analysis of variable elimination on graphs was the discovery of clique trees, that make the inherent parallelism in the elimination algorithm explicit [40, 59, 9]. The complexity of the numerical elimination step depends on the size of the largest clique, also called the tree width. For example, the variable ordering associated with the DAG in Fig. 9b splits the graph on the clique formed by \( F_2 \) and \( V_2 \). By taking advantage of this parallelism, we can solve this type of hybrid dynamics problem more efficiently. Nested dissection (ND) algorithms [25] try to exploit this by recursively partitioning the graph and returning a post-fix notation of the partitioning tree as the ordering.

D. Dynamics Problems for Closed Kinematic Loops

As discussed in [19], for inverse dynamics, if a manipulator with kinematic loop is redundantly actuated (the number of actuated joints is greater than the degree of motion freedom) there are infinitely many values of torque \( \tau \) that produce the same angular acceleration \( \dot{\theta} \). If a unique solution is required, one can either add more constraints or apply an optimality criterion, which can be done by adding extra factors to the graph. For example, minimum torque factors make the solution unique by choosing the minimum torque values. For forward dynamics, if a manipulator with kinematic loop is overconstrained (for example, any system containing planar kinematic loops), the constraint forces exerted by loop joints are underdetermined. We can convert this overconstrained system to be properly constrained if possible, for example, by replacing the original loop joint with a joint that imposes less constraints. With factor graphs, this can be done by adding a planar factor at the loop joint which reduces the number of unknown constraint forces so it can be properly solved.

In future work, we hope to apply these findings to perform kinodynamic motion planning in the style of GPMP2 [46] and STEAP [45], which successfully applied incremental inference in factor graphs [34, 15] to kinematic motion planning problems. In addition, it would be very interesting to use the factor-graph-based representation of dynamics to perform state estimation for dynamically balanced robots, in the spirit of Hartley et al. [31].

IV. DISCUSSION

In this paper, we represent manipulator dynamics as factor graph and solve for inverse, forward, and hybrid problems. Using factor graphs as a graphical language gives us not only a unified method to solve all types of dynamics problems, but also an insightful visualization of the underlying mathematical formulations. Exploiting different elimination orders of solving the factor graph unlocks powerful tools to illustrate classical algorithms and derive novel algorithms which could be applied to solve certain types of dynamics problems efficiently.

We use a five-bar parallel manipulator as shown in Fig. 10 to illustrate how to solve kinematic loops with factor graphs. This manipulator is properly actuated, since only joints 1 and 2 are actuated, and other joints are free to rotate. Joint 5 closes the loop. Link 0 represent the base link, which is fixed, and the end-effector is attached to link 2. Since

![Fig. 11: Five-bar parallel manipulator dynamics factor graph, where black dots represent variable constraints, namely factors, and circles represent variable nodes.](image-url)
[55] R. Seidel. A new method for solving constraint satisfaction problems. In *Intl. Joint Conf. on AI (IJCAI)*, pages 338–342, 1981.

[56] P. P. Shenoy and G. Shafer. Propagating belief functions using local computations. *IEEE Expert*, 1(3):43–52, Fall 1986.

[57] Y. Stepanenko and M. Vukobratovi. Dynamics of articulated open-chain active mechanisms. *Mathematical Biosciences*, 28(1-2):137–170, 1976.

[58] R. Tanner. A recursive approach to low complexity codes. *IEEE Trans. Inform. Theory*, 27(5):533–547, September 1981.

[59] R.E. Tarjan and M. Yannakakis. Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs and selectively reduce acyclic hypergraphs. *SIAM J. Comput.*, 13(3):566–579, 1984.

[60] M. W. Walker and D. E. Orin. Efficient dynamic computer simulation of robotic mechanisms. *Journal of Dynamic Systems, Measurement, and Control*, 104(3):205–211, 1982.