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

Mandy Xie and Frank Dellaert
School of Interactive Computing, Georgia Institute of Technology
Atlanta, Georgia 30332–0250, Email: {manxie,dellaert}@gatech.edu

*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 a 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 [43, 21] and similar methods described in [53] and [63]. 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) [67, 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 Section 9.2 of [19].

The methods mentioned above do not apply for manipulators with kinematic loops. Because of the complexity caused by kinematic redundancy [11], actuation redundancy [50], and uncertainty in constraint forces exerted by loop joints [19], more sophisticated and expensive algorithms are required to calculate their dynamics, which can be found in [57, 50] and Chapter 8 of [19]. More sophisticated and expensive algorithms are required to solve both inverse and forward dynamics for parallel robots, which can be found in [19, 57].

There is rarely a single algorithm which can solve all three types of dynamics problems. Different algorithms have to be applied as described in Section 5.3, 6.2, 7.3 of [19].

Rodriguez [58, 57] 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 [59]. Different algorithms have to be designed and implemented, though they share a unified framework. Based on the work by Rodriguez et al. [60], Jain [34] analyzed various algorithms for serial chain dynamics in a unified formulation. Ascher et al. [3] unified the derivation of CRBA and ABA as two elimination methods which are used to solve forward dynamics. Different algorithms have to be designed for each type [19], and they are not easy to be explained in an intuitive way. Rodriguez [58] used a random field estimation approach to solve both inverse and forward dynamics problems, which builds parallelism between the concepts of estimation used in Kalman filtering and smoothing theory and the dynamics problems. However, the parallelism is not straightforward and easy to visualize. Even though it is claimed in [57] that the same method can be used to solve dynamics for manipulators involving closed kinematic loop, non-trivial modifications have to be made for this method to be applied.

In this paper we introduce factor graphs as a unifying 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 factor graph 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.

Note that graphical models in general have been used before in robotic dynamics, e.g., Ting *et al*. [66] used Bayes networks to model system identification of rigid body parameters from noisy data. Factor graphs have also been used in walking robots by at least two different groups [32, 31, 68], but without explicit modeling of dynamical quantities as we do below.

We are also not the first ones to exploit ordering/permutations of matrices to improve performance: sparse linear algebra was already exploited very early on in [54] and is still being re-discovered regularly, e.g. [52]. Rather, the factor graph approach exposes a very general view on these problems and exposes elimination ordering in a much clearer way than typical sparse linear algebra methods, and opens up
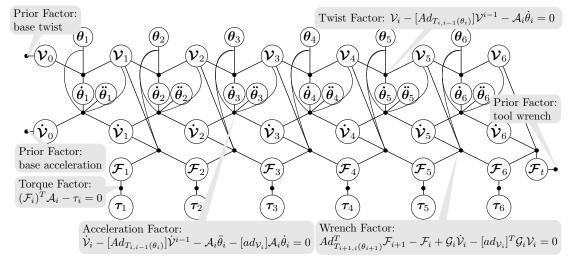
Fig. 1: The Puma 560 dynamics factor graph, where black dots represent factors, and circles represent variables.

the possibility of adding non-linearities into the same framework. Finally, the eliminated directed acyclic graphs (DAGs) (Section IV) point the way to automatically generating code corresponding to a topological ordering of the corresponding DAG.

## 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 [44]. As convincingly argued in their introduction, this geometric view pioneered by Brockett [9] and Murray et al. [49], 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 $f$ and torques $\tau$, can be expressed in body coordinates as (Equations 8.22 and 8.23 on page 242 in [44]),

$$f_b = m\dot{v}_b + \omega_b \times mv_b \quad (1)$$
$$\tau_b = \mathcal{I}_b\dot{\omega}_b + \omega_b \times \mathcal{I}_b\omega_b \quad (2)$$

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 (Equation 8.40 on page 247 of [44]),

$$\mathcal{F}_b = \mathcal{G}_b\dot{\mathcal{V}}_b - [ad_{\mathcal{V}_b}]^T\mathcal{G}_b\mathcal{V}_b \quad (3)$$

where the new quantities are defined as

$$\mathcal{V}_b = \begin{bmatrix} \omega_b \\ v_b \end{bmatrix} \quad \mathcal{F}_b = \begin{bmatrix} \tau_b \\ f_b \end{bmatrix}$$

$$\mathcal{G}_b = \begin{bmatrix} \mathcal{I}_b & 0 \\ 0 & mI \end{bmatrix} \quad [ad_{\mathcal{V}_b}] = \begin{bmatrix} [\omega_b] & 0 \\ [v_b] & [\omega_b] \end{bmatrix}$$

Above $[\omega_b]$ is the skew-symmetric matrix formed from $\omega_b$, i.e.,

$$[\omega_b] = R^T\dot{R}$$

with $R \in SO(3)$ the rotation associated with a link.

Closely following Section 8.3 in [44], 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 $\mathcal{V}_i$ and $\dot{\mathcal{V}}_i$ for the $i$-th link are expressed in a body-fixed coordinate frame rigidly attached to the link. The wrench transmitted through joint $i$ is denoted as $\mathcal{F}_i$, and $\mathcal{G}_i$ is the link's inertia matrix. Without loss of generality, below we assume all rotational joints, and we then have:

$$\mathcal{V}_i - [Ad_{T_{i,i-1}(\theta_i)}]\mathcal{V}_{i-1} - \mathcal{A}_i\dot{\theta}_i = 0 \quad (4)$$
$$\dot{\mathcal{V}}_i - [Ad_{T_{i,i-1}(\theta_i)}]\dot{\mathcal{V}}_{i-1} - \mathcal{A}_i\ddot{\theta}_i - [ad_{\mathcal{V}_i}]\mathcal{A}_i\dot{\theta}_i = 0 \quad (5)$$
$$Ad^T_{T_{i+1,i}(\theta_{i+1})}\mathcal{F}_{i+1} - \mathcal{F}_i + \mathcal{G}_i\dot{\mathcal{V}}_i - [ad_{\mathcal{V}_i}]^T\mathcal{G}_i\mathcal{V}_i = 0 \quad (6)$$
$$\mathcal{F}_i^T\mathcal{A}_i - \tau_i = 0 \quad (7)$$

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

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

Gravity is not considered above but can easily be accounted for. Lynch & Park [44] describe a standard "trick" that adds an extra acceleration term to the base. While clever, we have found it more intuitive to explicitly deal with gravity in our
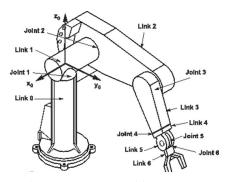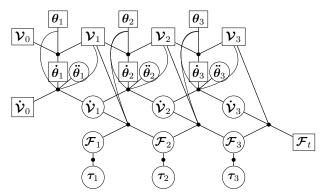
Fig. 2: The PUMA 560 robot [2].



Fig. 3: Dynamic factor graph for a 3R robot, with *known* variables shown as square nodes.



Fig. 4: (a) Simplified inverse dynamics factor graph (b) Block-sparse matrix corresponding to Fig. 4a

implementation, where we simply add a gravity term to the wrench equation (6), expressed in each link's body frame.

## III. A Factor Graph Approach

### A. Factor Graphs

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 [61, 23, 13], AI [55, 62, 38, 24, 37], sparse linear algebra [26, 29, 33], information theory [64, 42], combinatorial optimization [6, 5, 7], and even query theory [4, 17, 30]. A general theory specified in terms of algebraic semirings was also developed [10], and seminal work in theory proved essential computational complexity results [39] based on the existence of separator theorems for certain classes of graphs [40]. Factor graphs have been successfully applied in other areas of robotics, such as SLAM [36, 35, 15, 22], state estimation in humanoids [31], and (kinematic) motion planning [16, 48, 45].

### B. Dynamic Factor Graphs

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 1 illustrates this for the classical Puma 560 robot shown in Figure 2. Variables including twists $\mathcal{V}_i$, accelerations $\dot{\mathcal{V}}_i$, wrenches $\mathcal{F}_i$, joint angles $\theta_i$, joint velocities $\dot{\theta}_i$, joint accelerations $\ddot{\theta}_i$, and torques $\tau_i$. 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 $\theta_i$ and $\dot{\theta}_i$ are known for all joints. This in turn allows us to solve for the twist variables $\mathcal{V}_i$ in advance.

In addition, we typically also assume that $\dot{\mathcal{V}}_0$ and the end-effector wrench $\mathcal{F}_t$ 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. 3 for an RRR robot, which will be the starting point for the sections below.

### C. Automatic Transcription into a Factor Graph

The dynamic factor graphs for all results below are obtained automatically by converting Unified Robot Description Format (URDF) files programatically into an internal representation that works with the GTSAM factor graph library [14, 15]. This process is relatively straightforward, and the code will be released in the public domain. In essence, a bipartite graph of joints and links is created, which is then transcribed into joint-specific, link-specific, and joint-link interaction factors, as shown in Figure 1. For parallel robots, which are not supported by the URDF format out of the box, we provide the ability to provided an amended URDF file with extra loop closures, or parse a Simulation Description Format (SDF) file. Instructions for both formats are provided in the repository. To help with reviewing, an anonymized version of the code is available on https://anonymous.4open.science at this link.

## IV. 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.
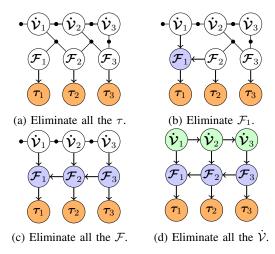
(a) Eliminate all the $\tau$.     (b) Eliminate $\mathcal{F}_1$.

(c) Eliminate all the $\mathcal{F}$.     (d) Eliminate all the $\dot{\mathcal{V}}$.

Fig. 5: Steps in the Elimination Algorithm.

---

**Algorithm 1:** Inverse dynamics corresponding to Fig. 5d.

1. $\dot{\mathcal{V}}_1 = [Ad_{T_{1,0}(\theta_1)}]\mathcal{V}_0 + [ad_{\mathcal{V}_1}]\mathcal{A}_1\dot{\theta}_1 + \mathcal{A}_1\ddot{\theta}_1$ ;

2. $\dot{\mathcal{V}}_2 = [Ad_{T_{2,1}(\theta_2)}]\dot{\mathcal{V}}_1 + [ad_{\mathcal{V}_2}]\mathcal{A}_2\dot{\theta}_2 + \mathcal{A}_2\ddot{\theta}_2$ ;

3. $\dot{\mathcal{V}}_3 = [Ad_{T_{3,2}(\theta_3)}]\dot{\mathcal{V}}_2 + [ad_{\mathcal{V}_3}]\mathcal{A}_3\dot{\theta}_3 + \mathcal{A}_3\ddot{\theta}_3$ ;

4. $\mathcal{F}_3 = Ad^T_{T_{t,3}}\mathcal{F}_t + \mathcal{G}_3\dot{\mathcal{V}}_3 - [ad_{\mathcal{V}_3}]^T\mathcal{G}_3\mathcal{V}_3$ ;

5. $\mathcal{F}_2 = Ad^T_{T_{3,2}(\theta_3)}\mathcal{F}_3 + \mathcal{G}_2\dot{\mathcal{V}}_2 - [ad_{\mathcal{V}_2}]^T\mathcal{G}_2\mathcal{V}_2$ ;

6. $\mathcal{F}_1 = Ad^T_{T_{2,1}(\theta_2)}\mathcal{F}_2 + \mathcal{G}_1\dot{\mathcal{V}}_1 - [ad_{\mathcal{V}_1}]^T\mathcal{G}_1\mathcal{V}_1$ ;

7. $\tau_1 = \mathcal{F}_1^T\mathcal{A}_1$ ;

8. $\tau_2 = \mathcal{F}_2^T\mathcal{A}_2$ ;

9. $\tau_3 = \mathcal{F}_3^T\mathcal{A}_3$ ;

---

For the RRR example, the resulting graph is shown in Fig. 4a, corresponding to the 9 linear constraints comprising the 3R inverse dynamics problem.

### A. Gaussian Elimination to a DAG

The Gaussian elimination algorithm to solve this set of linear equations can be graphically understood as converting the factor graph in Fig. 4a to a directed acyclic graph (DAG). Just as a factor graph is the graphical embodiment of the dynamics system, the DAG reveals the sparsity structure resulting after Gaussian elimination of the dynamics equations.

Elimination proceeds one variable at a time, and expresses that variable in terms of variables that will be eliminated later. If more than one equation is involved, this will result in new equations, i.e., factors, that can lead to so-called *fill-in* in the corresponding sparse system of linear equations. This graphical "elimination game" was first developed in sparse linear algebra [27, 29, 33], but is also used in probabilistic inference, where the DAG represents a Bayes net [55], and in sensor fusion problems in robotics [15].

We illustrate the elimination process in the 3R case for a particular ordering in Fig. 5. The elimination is performed in the order $\tau_3 \ldots \tau_1$, $\mathcal{F}_1 \ldots \mathcal{F}_3$, $\dot{\mathcal{V}}_3 \ldots \dot{\mathcal{V}}_1$. Fig. 5a shows the result of first eliminating the torques $\tau_i$, where the arrows show that the torques $\tau_i$ only depend on the corresponding wrenches $\mathcal{F}_i$. Fig. 5b shows the elimination of $\mathcal{F}_1$, which results in a dependence of $\mathcal{F}_1$ on $\dot{\mathcal{V}}_1$ and $\mathcal{F}_2$. After eliminating all the wrenches, we get the result as shown in Fig. 5c. Finally, we eliminate the all the twist accelerations $\dot{\mathcal{V}}_3 \ldots \dot{\mathcal{V}}_1$, in that order. After completing all these elimination steps, the inverse dynamics factor graph in Fig. 4a is thereby converted to the DAG as shown in Fig. 5d.

### B. 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, it it matters which variables are eliminated

first. For example, eliminating $\tau_3$ above is simply a matter of rewriting (7)

$$\mathcal{F}_3^T\mathcal{A}_3 - \tau_3 = 0$$

as

$$\tau_3 = \mathcal{F}_3^T\mathcal{A}_3$$

However, if one were to eliminate $\mathcal{F}_2$ in Fig. 4a first, it would involve three constraints (the number of factors attached to $\mathcal{F}_2$) 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 $\dot{\mathcal{V}}_i$, link wrenches $\mathcal{F}_i$, and then the scalar torques. The back-substitution sequence corresponding to Fig. 5d 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) [43]. The resulting DAG can be viewed as a graphical representation of RNEA, where the green and blue/orange colors resp. correspond to the forward path and the backward path.

### C. 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 $6 \times 6$, except where the (scalar) torques $\tau_i$ are concerned. For example, Fig 4b shows the sparse block-matrix corresponding to the simplified 3R inverse dynamics

TABLE I: Numerical inverse dynamics for a PUMA 560

| Elimination Method | Average Time($\mu s$) |
|---|---|
| RNEA | 26.6 |
| RNEA in RBDL | 20.2 |
| COLAMD | 11.0 |
| ND | 11.7 |



(a)              (b)

Fig. 6: (a) Simplified forward dynamics graph and (b) corresponding block-sparse matrix.

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 |

graph in Fig. 4a. 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 $\dot{\mathcal{V}}_i$, 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 [1] and nested dissection (ND) [25], which are two state of the art sparse linear algebra ordering heuristics.
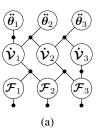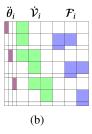
The reported results were obtained using GTSAM [14], a general factor graph solver used extensively in the robotics community. However, we make no claim that these results in any way come close to dedicated dynamics solvers, and they are intended to indicate relative performance rather than claiming SOA absolute performance. GTSAM is a very general library which is optimized towards much larger problems. In addition, once an ordering is chosen we should be able to perform symbolic elimination only once, rather than doing it every time as in our results. As alluded to above, in future work we plan to automatically generate code for particular robot topologies, which we hypothesize will match and possibly exceed existing solvers when using non-intuitive but computationally more advantageous elimination orderings.

### D. 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.

### V. 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. 6a.

### A. Solving Symbolically

In very much the same spirit as our work, Ascher et al. [3] 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. 7a and Fig. 7b. CRBA can be explained as first eliminating all the wrenches $\mathcal{F}_i$, then eliminating all the accelerations $\dot{\mathcal{V}}_i$, and lastly eliminating all the angular accelerations $\ddot{\theta}_i$. In contrast, in ABA we alternate eliminating the wrenches $\mathcal{F}_i$, accelerations $\dot{\mathcal{V}}_i$ 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 in terms of operation counts. Indeed, different elimination orderings result in different fill-in, and an ordering with minimum fill-in minimizes the cost of the elimination algorithm![15]. For example, from Figures 7a and 7b 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. [3], but in the graphical framework we can tell this directly by observing the DAG.

### B. 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].

(a) CRBA ordering.    (b) ABA ordering.    (c) COLAMD ordering.    (d) ND ordering.
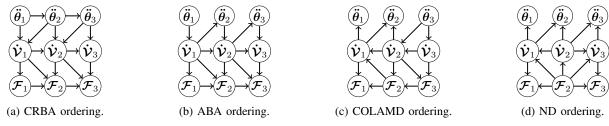
Fig. 7: DAGs generated by solving forward dynamics factor graph with different variable elimination orderings.
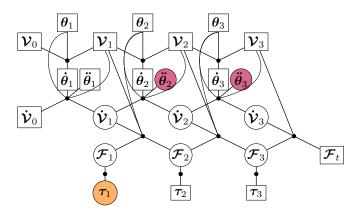

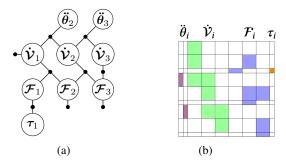
Fig. 8: Hybrid dynamics factor graph for a 3R robot.



Fig. 9: (a) Simplified hybrid dynamics graph and (b) corresponding block-sparse.

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 ABA indeed outperforms CRBA in this case. However, the two sparse linear algebra ordering heuristics COLAMD and ND outperform both, by yet another factor of two or more.

## VI. HYBRID DYNAMICS

In hybrid dynamics problems, either $\ddot{\theta}_i$ 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 Section 9.2 of [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".
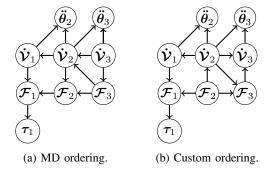


(a) MD ordering.    (b) Custom ordering.

Fig. 10: DAGs from solving hybrid dynamics factor graph.

### A. Featherstone's method

Solving the hybrid dynamics using Featherstone's algorithm can be illustrated with factor graphs using a simple 3R example. Fig. 8 shows a factor graph for the case when $\tau_1$ is unknown while $\ddot{\theta}_1$ is given, and additionally $\ddot{\theta}_2$ and $\ddot{\theta}_3$ 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. 9a.

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

### B. 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. 8. Using an elimination variable ordering generated by a minimum degree (MD) heuristic shown in Table III, the resulting DAG obtained is shown in Fig. 10a. For good measure, we also eliminated the factor graph with another, manually created elimination order

TABLE III: Elimination Orders for Hybrid Dynamics

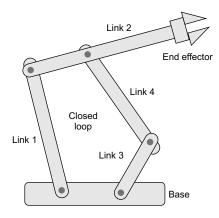| Elimination Method | Elimination Order |
|---|---|
| MD | $t_1, a_2, a_3, \dot{\mathcal{V}}_3, \mathcal{F}_1, \dot{\mathcal{V}}_1, \mathcal{F}_2, \dot{\mathcal{V}}_2, \mathcal{F}_3$ |
| CUSTOM | $t_1, a_2, a_3, \mathcal{F}_1, \dot{\mathcal{V}}_1, \dot{\mathcal{V}}_3, \mathcal{F}_3, \dot{\mathcal{V}}_2, \mathcal{F}_2$ |



Fig. 11: Five-bar parallel manipulator (adapted from [56]). The bottom 2 joints are actuated but other joints move freely.

listed as "CUSTOM" in Table III, and show the corresponding DAG in Fig. 10b. The two DAGs are slightly different, but both have the same number of directed edges and hence might be suspected to have the same computational complexity.

However, especially for hybrid problems like this, being sophisticated about variable ordering and the possible resulting parallelism 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 [41, 65, 8]. A clique tree or directed Bayes tree [35] can be constructed from the DAG to guide parallel execution. The complexity of the numerical elimination step depends on the *tree width*, i.e., the size of the largest clique in the tree. For example, the variable ordering associated with the DAG in Fig. 10b splits the graph on the clique formed by $\mathcal{F}_2$ and $\dot{\mathcal{V}}_2$. 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.

## VII. Dynamics with Closed Kinematic Loops

As discussed in Chapter 8 of [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 $\ddot{\theta}$. 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.

We use a five-bar parallel manipulator as shown in Fig. 11 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 the kinematic loop in this manipulator is planar, we add a planar factor to the dynamics factor graph at the loop joint as shown in Fig. 12, where the planar factor is shown as a unary factor associated with wrench $\mathcal{F}_5$, which is the unknown constraint wrench exerted by the loop joint. With the closed loop dynamics factor graph, we can solve inverse, forward and hybrid dynamics problems by specifying which variables are known and which are unknown. We can solve this factor graph with any elimination ordering, and the resulting DAGs can be taken as graphical representations for different algorithms to solve closed-loop problems.

## VIII. 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.

As we discussed Section IV, the reported timing results are in no way intended to compete with dedicated dynamics solvers, but rather indicated relative performance. In future work we plan to automatically generate code for particular robot topologies, which we hypothesize will match and possibly exceed existing solvers when using non-intuitive but computationally more advantageous elimination orderings. We are also aware that in comparing high performance code controlling for cache effects and memory architecture in general is important, as done by Neuman *et al.* [51].

In future work, we hope to apply these findings to perform kinodynamic motion planning in the style of GPMP2 [47] and STEAP [46], which successfully applied incremental inference in factor graphs [35, 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.* [32, 31] and Wisth *et al.* [68].
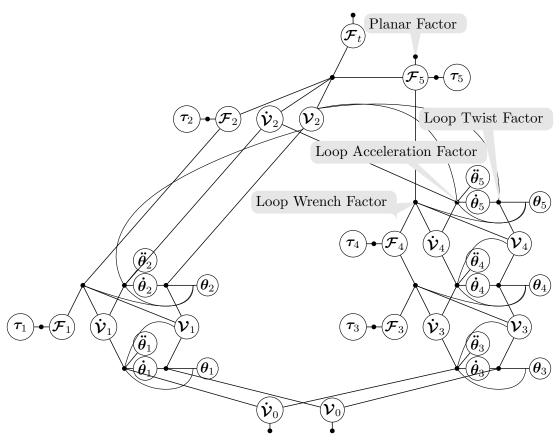
Fig. 12: Five-bar parallel manipulator dynamics factor graph, where black dots represent variable constraints, namely factors, and circles represent variable nodes.

## REFERENCES

[1] P.R. Amestoy, T. Davis, and I.S. Duff. An approximate minimum degree ordering algorithm. *SIAM Journal on Matrix Analysis and Applications*, 17(4):886–905, 1996.

[2] B. Armstrong, O. Khatib, and J. Burdick. The explicit dynamic model and inertial parameters of the puma 560 arm. In *Proceedings. 1986 IEEE international conference on robotics and automation*, volume 3, pages 510–518. IEEE, 1986.

[3] U. M. Ascher, P. K. Dinesh, and B. P. Cloutier. Forward dynamics, elimination methods, and formulation stiffness in robot simulation. *The International Journal of Robotics Research*, 16(6):749–758, 1997.

[4] C. Beeri, R. Fagin, D. Maier, A. Mendelzon, J. Ullman, and M. Yannakakis. Properties of acyclic database schemes. In *ACM Symp. on Theory of Computing (STOC)*, pages 355–362, New York, NY, USA, 1981. ACM Press.

[5] U. Bertele and F. Brioschi. On the theory of the elimination process. *J. Math. Anal. Appl.*, (1):48–57, July.

[6] U. Bertele and F. Brioschi. *Nonserial Dynamic Programming*. Academic Press, 1972.

[7] U. Bertele and F. Brioschi. On nonserial dynamic programming. *J. Combinatorial Theory*, 14:137–148, 1973.

[8] J.R.S. Blair and B.W. Peyton. An introduction to chordal graphs and clique trees. In George et al. [28], pages 1–27.

[9] W. Roger Brockett. Robotic manipulators and the product of exponentials formula. In *Mathematical theory of networks and systems*, pages 120–129. Springer, 1984.

[10] B. A. Carré. An algebra for network routing problems. *J. Inst. Math. Appl.*, 7:273–294, 1971.

[11] S. Chiaverini, G. Oriolo, and I. D. Walker. Kinematically redundant manipulators. *Springer handbook of robotics*, pages 245–268, 2008.

[12] J. J Craig. *Introduction to robotics: mechanics and control, 3/E*. Pearson Education India, 2009.

[13] R. Dechter and J. Pearl. Network-based heuristics for constraint-satisfaction problems. *Artificial Intelligence*, 34(1):1–38, December 1987.

[14] F. Dellaert. Factor graphs and GTSAM: A hands-on introduction. Technical Report GT-RIM-CP&R-2012-002, Georgia Institute of Technology, 2012.

[15] F. Dellaert and M. Kaess. Factor graphs for robot perception. *Foundations and Trends in Robotics*, 6(1-2):1–139, 2017.

[16] J. Dong, M. Mukadam, F. Dellaert, and B Boots. Motion planning as probabilistic inference using Gaussian

processes and factor graphs. In *Robotics: Science and Systems (RSS)*, 2016.

[17] R. Fagin, A.O. Mendelzon, and J.D. Ullman. A simplied universal relation assumption and its properties. *ACM Trans. Database Syst.*, 7(3):343–360, 1982.

[18] R. Featherstone. The calculation of robot dynamics using articulated-body inertias. *The International Journal of Robotics Research*, 2(1):13–30, 1983.

[19] R. Featherstone. *Rigid body dynamics algorithms*. Springer, 2014.

[20] R. Featherstone and D. E. Orin. Robot dynamics: equations and algorithms. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 1, pages 826–834. IEEE, 2000.

[21] M. L. Felis. RBDL: An efficient rigid-body dynamics library using recursive algorithms. *Autonomous Robots*, 41(2):495–511, 2017.

[22] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza. On-manifold preintegration for real-time visual-inertial odometry. *IEEE Trans. Robotics*, 2017.

[23] Eugene C. Freuder. A sufficient condition for backtrack-free search. *J. ACM*, 29(1):24–32, 1982.

[24] B.J. Frey, F.R. Kschischang, H.-A. Loeliger, and N. Wiberg. Factor graphs and algorithms. In *Proc. 35th Allerton Conf. Communications, Control, and Computing*, pages 666–680, September 1997.

[25] A. George. Nested dissection of a regular finite element mesh. *SIAM Journal on Numerical Analysis*, 10(2):345–363, April 1973.

[26] A. George, J. Liu, and Ng E. Row-ordering schemes for sparse Givens transformations. I. Bipartite graph model. *Linear Algebra Appl*, 61:55–81, 1984.

[27] J.A. George, J.R. Gilbert, and J.W-H. Liu, editors. *Graph Theory and Sparse Matrix Computations*, volume 56 of *IMA Volumes in Mathematics and its Applications*. Springer-Verlag, 1993.

[28] J.A. George, J.R. Gilbert, and J.W-H. Liu, editors. *Graph Theory and Sparse Matrix Computations*, volume 56 of *IMA Volumes in Mathematics and its Applications*. Springer-Verlag, New York, 1993.

[29] J.R. Gilbert and E.G. Ng. Predicting structure in non-symmetric sparse matrix factorizations. In George et al. [28].

[30] N. Goodman and O. Shmueli. Tree queries: a simple class of relational queries. *ACM Trans. Database Syst.*, 7(4):653–677, 1982.

[31] R. Hartley, M. G. Jadidi, L. Gan, J. Huang, J. W. Grizzle, and R. M. Eustice. Hybrid contact preintegration for visual-inertial-contact state estimation using factor graphs. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3783–3790, Oct 2018.

[32] R. Hartley, J. Mangelson, L. Gan, M. Ghaffari Jadidi, J. M. Walls, R. M. Eustice, and J. W. Grizzle. Legged robot state-estimation through combined forward kine-

[33] matic and preintegrated contact factors. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4422–4429, May 2018.

[33] P. Heggernes and P. Matstoms. Finding good column orderings for sparse QR factorization. In *Second SIAM Conference on Sparse Matrices*, 1996.

[34] A. Jain. Unified formulation of dynamics for serial rigid multibody systems. *Journal of Guidance, Control, and Dynamics*, 14(3):531–542, 1991.

[35] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. Leonard, and F. Dellaert. iSAM2: Incremental smoothing and mapping using the Bayes tree. *Intl. J. of Robotics Research*, 31:217–236, Feb 2012.

[36] M. Kaess, A. Ranganathan, and F. Dellaert. iSAM: Incremental smoothing and mapping. *IEEE Trans. Robotics*, 24(6):1365–1378, Dec 2008.

[37] F.R. Kschischang, B.J. Frey, and H-A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Trans. Inform. Theory*, 47(2), February 2001.

[38] S. L. Lauritzen and D. J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society. Series B (Methodological)*, 50(2):157–224, 1988.

[39] R.J. Lipton, D.J. Rose, and R.E. Tarjan. Generalized nested dissection. *SIAM Journal on Applied Mathematics*, 16(2):346–358, 1979.

[40] R.J. Lipton and R.E. Tarjan. A separator theorem for planar graphs. *SIAM Journal on Applied Mathematics*, 36(2):177–189, April 1979.

[41] J. W-H. Liu and A. Mirzaian. A linear reordering algorithm for parallel pivoting of chordal graphs. *SIAM J. Disc. Math.*, 2:lOO–107, 1989.

[42] H.-A. Loeliger. An introduction to factor graphs. *IEEE Signal Processing Magazine*, pages 28–41, January 2004.

[43] J. Y. S. Luh, M. W. Walker, and R. P. Paul. On-line computational scheme for mechanical manipulators. *Journal of Dynamic Systems, Measurement, and Control*, 102(2):69–76, 1980.

[44] K. M Lynch and F. C Park. *Modern Robotics*. Cambridge University Press, 2017.

[45] M. Mukadam, J. Dong, F. Dellaert, and B. Boots. Simultaneous trajectory estimation and planning via probabilistic inference. In *Robotics: Science and Systems (RSS)*, 2017.

[46] M. Mukadam, J. Dong, F. Dellaert, and B. Boots. Steap: simultaneous trajectory estimation and planning. *Autonomous Robots*, pages 1–20, 2018.

[47] M. Mukadam, J. Dong, X. Yan, F. Dellaert, and B. Boots. Continuous-time gaussian process motion planning via probabilistic inference. *Intl. J. of Robotics Research*, 37(11):1319–1340, 2018.

[48] M. Mukadam, X. Yan, and B. Boots. Gaussian process motion planning. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2016.

[49] R.M. Murray, Z. Li, and S. Sastry. *A Mathematical*

*Introduction to Robotic Manipulation*. CRC Press, 1994.

[50] Y. Nakamura and M. Ghodoussi. Dynamics computation of closed-link robot mechanisms with nonredundant and redundant actuators. *IEEE Transactions on Robotics and Automation*, 5(3):294–302, 1989.

[51] S. M. Neuman, T. Koolen, J. Drean, J. E. Miller, and S. Devadas. Benchmarking and workload analysis of robot dynamics algorithms. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5235–5242, Nov 2019.

[52] F. Nori. Inverse, forward and other dynamic computations computationally optimized with sparse matrix factorizations. In *2017 IEEE International Conference on Real-time Computing and Robotics (RCAR)*, pages 371–377, July 2017.

[53] D. E. Orin, R. McGhee, M. Vukobratovi, and G. Hartoch. Kinematic and kinetic analysis of open-chain linkages utilizing newton-euler methods. *Mathematical Biosciences*, 43(1-2):107–130, 1979.

[54] N Orlandea and DA Calahan. A sparsity-oriented approach to the design of mechanical systems. *Problem Analysis in Science and Engineering*, pages 361–389, 1977.

[55] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[56] Reddit Robotics. 5-bar parallel manipulator.

[57] G. Rodriguez. Recursive forward dynamics for two robot arms in a closed chain based on kalman filtering and bryson-frazier smoothing. In *Proceedings of the International Symposium on Robot Manipulators on Recent trends in robotics: modeling, control and education*, pages 85–93. Elsevier North-Holland, Inc., 1986.

[58] G. Rodriguez. Kalman filtering, smoothing, and recursive robot arm forward and inverse dynamics. *IEEE Journal on Robotics and Automation*, 3(6):624–639, 1987.

[59] G. Rodriguez. Recursive forward dynamics for multiple robot arms moving a common task object. *IEEE Transactions on Robotics and Automation*, 5(4):510–521, 1989.

[60] G. Rodriguez, A. Jain, and K. Kreutz-Delgado. A spatial operator algebra for manipulator modeling and control. *The International Journal of Robotics Research*, 10(4):371–381, 1991.

[61] R. Seidel. A new method for solving constraint satisfaction problems. In *Intl. Joint Conf. on AI (IJCAI)*, pages 338–342, 1981.

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

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

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

[65] 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.

[66] J-A Ting, M. Mistry, J. Peters, S. Schaal, and J. Nakanishi. A bayesian approach to nonlinear parameter identification for rigid body dynamics. In *Robotics: Science and Systems*, pages 32–39. Philadelphia, USA, 2006.

[67] 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.

[68] D. Wisth, M. Camurri, and M. Fallon. Robust legged robot state estimation using factor graph optimization. *IEEE Robotics and Automation Letters*, 4(4):4507–4514, Oct 2019.