Space Operator Algebra (SOA) notation for multibody dynamics

SOA was invented by Guillermo Rodriguez at JPL in the mid ‘80s for robotics research. He recognized the rich mathematical parallels between the Kalman filtering in the time domain and multibody dynamics in the spatial domain. Kalman methods are lower cost and numerically more accurate compared to traditional methods (carries over to multibody dynamics as well!). Led to the SOA methodology that provides the mathematical language for making minimal coordinate dynamics tractable. Modernized and made standard the “spatial” vector/inertia conventions widely used now.

Guy Man convinced the Cassini project on the ability of SOA algorithms to speed up flex dynamics simulations – leading to initial DARTS multibody dynamics software. DARTS has been subsequently extended and applied to robotics, landers, ground vehicles, rotorcraft, molecular dynamics applications. SOA has been significantly generalized and extended over the years. SOA book published in 2011.

Space vector notation SOA is a mathematical framework for studying and analyzing multibody dynamics exploiting its structure for low-cost recursive (minimal coordinate) algorithms. SOA theory originated at JPL (Jet Propulsion Laboratory), and software used for several research and mission applications. Lots of publications, but SOA is not taught in academia, and its methods are not well understood. Here we describe basic technical ideas towards developing a working knowledge of the SOA methodology from focusing on SOA foundations, explore new topics and areas. Most of topics covered there follow dissertation of Jain on SOA as presented in https://dartslab.jpl.nasa.gov/ (dated 2024), to follow state of the art.

Space vector notation has encountered increasing attention in recent years respect more traditional treatment of rigid body dynamics because of its suitability, since more compact and clear, making algorithms less complex easy to understand. In addition this formalism is very useful in multibody dynamics since allows a quite straightforward way for managing multibody systems models equations development.

Analytical advantages

- General: Directly applies to large class of systems – rigid/flex, serial/arbitrary tree topology, small large
- Also applies to closed-chain systems via constraint embedding (graph transformation)
- Provides a unifying language and abstract framework for tackling the complexities of dynamics
- Can do old things better: fast recursive algorithms (optimal, better numerical behavior)
- Can handle changes to topology easily: the structure remains the same, just continue with new topology!
- Can do new things: OSC techniques, mass matrix inverse, Fixman potential, diagonalized dynamics, inter-body forces

Application advantages

- Provides fast O(N) family of computational algorithms (optimal cost) – for simulation and embedded use
- Structure based – hence can accommodate changes to system topology
- Expressive – hence can accommodate large and diverse family of computations within single setting ideal for robotics and other applications
- Applies to broad class of system topologies – and hence can cover very broad needsApplies to rigid and non-rigid bodies, and hence allows for high fidelity physics
- Forms basis for PyCraft workbench for system level properties
- Operator structure carries over to subgraphs allowing the restriction of computational algorithms to smaller sub-systems as needed
- Math foundation allows novel algorithms as needed
- Provides a model-based computational architecture ideal for robotics – variable topology, constraints, task activities, control

Use SOA to tackle general multibody systems

- rigid/flex bodies
- arbitrary size and branched topologies
- closed-chain topologies

It turns out that the SOA methods developed for the serial chain case carry over virtually entirely to the broader class of multibody systems.

Minimal coordinates

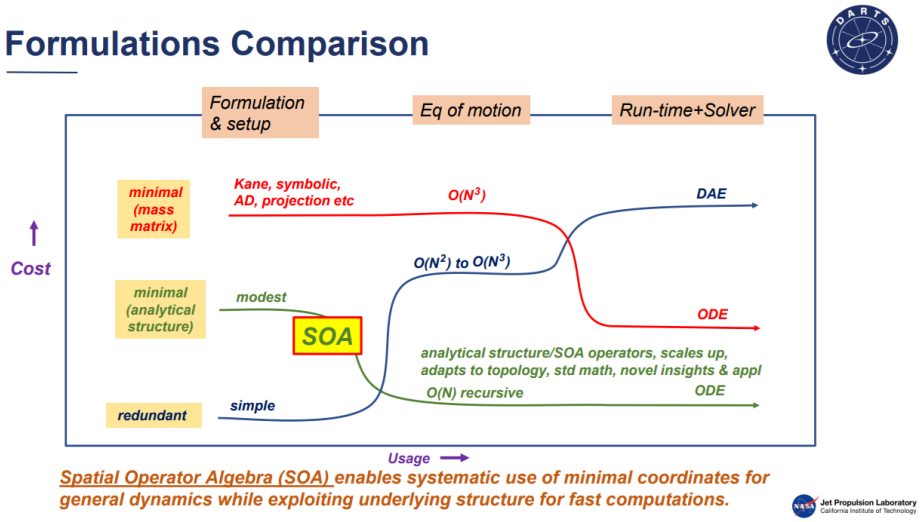
Minimal/Relative coordinates approach is characterized by

- Minimal hinge coordinates, ODE form
- Smaller no. of dofs
- Dense, configuration dependent mass matrix

Absolute/Redundant coordinates approach is characterized by:

- All bodies are treated as independent
- Bilateral constraints used for all hinges, DAE form
- Large no. of dofs, but simple to set up
- Constant, block diagonal mass matrix

In picture a short comparison is given between advantages of different approaches:



- **Minimal coordinate**: while complex, have smaller models, ODE formulation & rich underlying structure
- **Operators**: SOA's expressive mathematical language for 'analytical multibody dynamics' & exploiting this structure • Breadth: for concisely representing large family of key dynamics & kinematics quantities (Jacobian, mass matrix, OSC etc)
- **Depth**: Operator expressions remain invariant to size, branching topology, rigid/flex bodies, constraint embedding!
- **Algebra**: Can combine and manipulate operator expressions to get simpler expressions
- **Computation**: Can directly operator expressions into fast "O(N)" structure based recursive algorithms by simple inspection

Spatial Notation basics

When working with kinematics and dynamics, we often have to work with a combination of linear and angular properties: position/attitude, linear/angular velocities, force/moments, linear/angular momentum, mass/inertia.
Only at the body CM are the position & angular properties mostly decoupled. However often have to work with general, nonCM reference frames – get messy coupling.
Spatial" notation combines linear & angular terms together to help work with general (and not just CM) frames

- position/attitude: homogeneous transform
- velocities: spatial velocity [w, v] 6-dimensional
- accelerations: like-wise
- forces: spatial force [N, F] 6-dimensional

Not to be confused with twists/wrenches from classical kinematics theory, or with "spatial operators" coming up later

Propagation relationships are building blocks for recursively computing body/node properties for articulated multibody systems

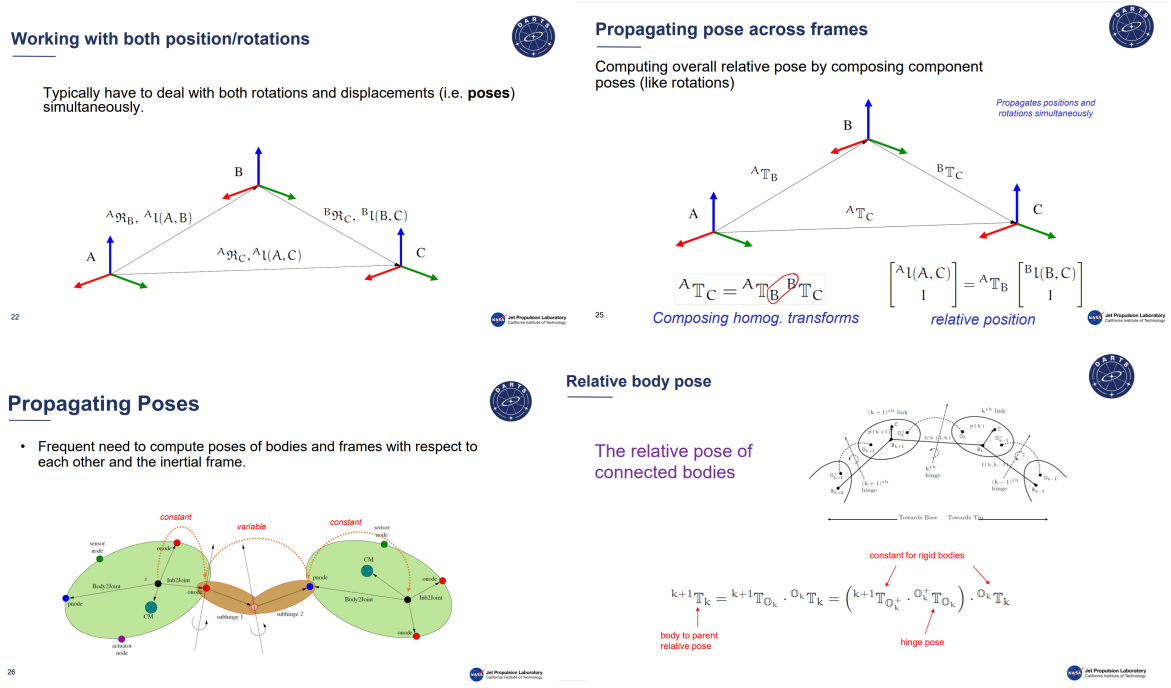
The * symbol is used to denote matrix transpose: $A^{\star} = A^{T}$. The ~ symbol denotes the cross-product matrix, as in other conventions.

Basic quantities involved in SOA:

- Spatial velocities (as spatial 6 vector)
- Spatial accelerations (as spatial 6 vector)
- Spatial forces (as spatial 6 vector)
- Spatial momentum
- Kinetic energy
- Spatial inertia (as spatial 6x6 2 tensor)

Spatial notation offers concise and consistent transformation expressions for arbitrary non-CM points, can transform spatial quantities across any pair of points.
Common choices for the H frame are:

- the inertial frame I
- the “from” frame F
- the “to” frame G



Connections with Lie group: Rotations form the SO3 Lie group. Homogenous transforms form the SE3 Lie group. Spatial velocities defined here are closely related to (but not the same as) left/right trivialization elements of the Lie algebra. The spatial cross product is the Lie bracket (commutator) operator corresponds to the Ad adjoint transformations for the SE3 Lie group. Mentioning these connections is for completeness, but these will not be essential to our development.

Kinetic energy is ***invariant*** to reference point when working with spatial quantities (using galilean map is not).

$$\mathcal{K}_e = \frac{1}{2}\mathcal{V}^\star(x)M(x)\mathcal{V}(x) = \frac{1}{2}\mathcal{V}^\star(y)M(y)\mathcal{V}(y)$$

This is a ***generalization*** of the well know quadratic expression for linear and angular energies at the CM. The total kinetic energy of the serial chain, or in general of a multibody system, Ke, is the sum of the kinetic energies of each of the individual links:

$$\mathcal{K}_e = \sum_{k=1}^{n} \frac{1}{2} \mathcal{V}^\star(k) M(k) \mathcal{V}(k)$$

Equation can be re-expressed as:

$$\mathcal{K}_e = \frac{1}{2} \begin{pmatrix} \mathcal{V}^\star(1) & \dots & \mathcal{V}^\star(1) \end{pmatrix} \begin{pmatrix} M(1) & 0 & \dots & 0 \\ 0 & M(2) & 0 & \vdots \\ \vdots & 0 & \ddots & 0 \\ 0 & \dots & 0 & M(n) \end{pmatrix} \begin{pmatrix} \mathcal{V}(1) \\ \vdots \\ \mathcal{V}(n) \end{pmatrix} = \frac{1}{2} \mathcal{V}^\star \mathbf{M} \mathcal{V}$$

where **M** is the block-diagonal, symmetric and positive semi-definite **spatial inertia spatial operator** defined as

.Substituting $\mathcal{V} = \Phi^\star H^\star \dot{\theta}$

$$\mathcal{K}_e = \frac{1}{2} \dot{\theta}^\star H \Phi M(k) H^\star \Phi^\star \dot{\theta} = \frac{1}{2} \dot{\theta}^\star \mathcal{M} \dot{\theta}$$

$$\mathcal{M} = H \Phi M(k) H^\star \Phi^\star \in \mathcal{R}^{\mathcal{N} \times N}$$

This factored form of the mass matrix is referred to as the **Newton–Euler Operator Factorization** of the mass matrix.

Spatial inertia is defined consequently to spatial vectors

$$M(x) = \begin{pmatrix} \mathcal{I}(x) & m\tilde{p}(x) \\ -m\tilde{p}(x) & mI_3 \end{pmatrix}$$

Parallel axis theorem allows one to transform inertia properties from one body reference point to another

- • Plain 3x3 rigid body inertia from CM inertia
- • Inertia transformation from arbitrary point y to point x is more involved

Often need total effective mass properties of a collection of bodies requires transforming all mass properties to a common point (parallel axis theorem) and then summing them up.

While the expressions are compact and concise, most of them involve sparse terms, and can optimize implementations for speed.

To define kinematics some fundamental quantities are used

- H is known as hinge map and convert generalized coordinates into 6-spatial vectors
- ϕ frame transformation matrix
- V is space 6-velocity
- α is space 6-acceleration
- f is space 6-force

Frame rigid transformation matrix

Rigid body transformation matrix Φ (6x6) spatial velocity propagation uses components of the T(B,C) homogeneous transform.

$$\Phi\left(B,C\right)=\begin{pmatrix}I_3 & \tilde{l}\left(B,C\right)\\ 0_{3x3} & I_3\end{pmatrix}$$

$$\Phi\left(B,C\right)=\begin{pmatrix}I_3 & \tilde{l}\left(B,C\right)\\ 0_{3x3} & I_3\end{pmatrix}\begin{pmatrix}R_3 & 0_{3x3}\\ 0_{3x3} & R_3\end{pmatrix}$$

It is applicable obviously also for treating kinematics between rigid bodies.
The space operator define essentially the geometrical configuration of the system and consequently jacobian structure. The rigid body transformation matrix from parent to child body of a serial chain is a classic example of. Note this is configuration dependent! Φ is involved also in velocity propagation across hinges, where a spatial velocity on the inboard side of the hinge is combined with hinge contribution to give spatial velocity on the outboard side of the hinge.

Rigid body transformation matrix satisfy properties

$$\Phi\left(x,x\right)=I_6$$
$$\Phi\left(x,z\right)=\Phi\left(x,y\right)\Phi\left(y,z\right)$$
$$\Phi^{-1}\left(x,y\right)=\Phi\left(y,x\right)$$

These equations are of

Hinge map matrix

Hinge map matrix is used as an operator able to act on spatial 6-vector
Time derivative of the hinge pose

The joint map matrix maps the (non-dimensional) hinge generalized velocities to the relative hinge spatial velocity

Can partition into angular/linear parts

configuration dofs can exceed velocity dofs

For a full 6dof hinge, the hinge map matrix is the identity matrix.

Table 2.5 Joint model formulas for one-degree-of-freedom lower pair joints, with abbreviations $c_{\theta_i} := \cos\theta_i$ and $s_{\theta_i} := \sin\theta_i$ (adapted in part from Table 4.1 in [2.22])

Joint type	Joint rotation matrix ${}^jR_i$	Position vector ${}^jp_i$	Free modes Φ_i	Constrained modes Φ_i^c	Pose state vars.	$\dot{q}_i$
Revolute R	$\begin{pmatrix} c_{\theta_i} & -s_{\theta_i} & 0 \\ s_{\theta_i} & c_{\theta_i} & 0 \\ 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$	θ_i	$\dot{\theta}_i$
Prismatic P	$1_{3\times3}$	$\begin{pmatrix} 0 \\ 0 \\ d_i \end{pmatrix}$	$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$	d_i	$\dot{d}_i$
Helical H (pitch h)	$\begin{pmatrix} c_{\theta_i} & -s_{\theta_i} & 0 \\ s_{\theta_i} & c_{\theta_i} & 0 \\ 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 0 \\ h\theta_i \end{pmatrix}$	$\begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ h \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -h \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$	θ_i	$\dot{\theta}_i$

Table 2.6 Joint model formulas for higher-degree-of-freedom lower pair joints, universal joint, rolling contact joint, and 6-DOF joint, with abbreviations $c_{\theta_i} := \cos\theta_i$ and $s_{\theta_i} := \sin\theta_i$ (adapted in part from Table 4.1 in [2.22]) *The Euler angles α_i, β_i, and γ_i could be used in place of the unit quaternion ϵ_i to represent orientation

Joint type	Joint rotation matrix ${}^jR_i$	Position vector ${}^jp_i$	Free modes Φ_i	Constrained modes Φ_i^c	Pose Variables	Velocity Variables $\dot{q}_i$
Cylindrical C	$\begin{pmatrix} c_{\theta_i} & -s_{\theta_i} & 0 \\ s_{\theta_i} & c_{\theta_i} & 0 \\ 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 0 \\ d_i \end{pmatrix}$	$\begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$	θ_i d_i	$\begin{pmatrix} \dot{\theta}_i \\ \dot{d}_i \end{pmatrix}$
Spherical* S	$\begin{pmatrix} \text{Table 2.1} \end{pmatrix}$	$\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$	ϵ_i	$\omega_{i\,\mathrm{rel}}$
Planar	$\begin{pmatrix} c_{\theta_i} & -s_{\theta_i} & 0 \\ s_{\theta_i} & c_{\theta_i} & 0 \\ 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} c_{\theta_i} d_{xi} - s_{\theta_i} d_{yi} \\ s_{\theta_i} d_{xi} + c_{\theta_i} d_{yi} \\ 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$	θ_i d_{xi} d_{yi}	$\begin{pmatrix} \dot{\theta}_i \\ \dot{d}_{xi} \\ \dot{d}_{yi} \end{pmatrix}$
Flat planar rolling contact (fixed radius r)	$\begin{pmatrix} c_{\theta_i} & -s_{\theta_i} & 0 \\ s_{\theta_i} & c_{\theta_i} & 0 \\ 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} r\theta_i c_{\theta_i} - r s_{\theta_i} \\ -r\theta_i s_{\theta_i} - r c_{\theta_i} \\ 0 \end{pmatrix}$	$\begin{pmatrix} 0 \\ 0 \\ 1 \\ r \\ 0 \\ 0 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -r & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$	θ_i	$\dot{\theta}_i$
Universal U	$\begin{pmatrix} c_{\alpha_i} c_{\beta_i} & -s_{\alpha_i} & c_{\alpha_i} s_{\beta_i} \\ s_{\alpha_i} c_{\beta_i} & c_{\alpha_i} & s_{\alpha_i} s_{\beta_i} \\ -s_{\beta_i} & 0 & c_{\beta_i} \end{pmatrix}$	$\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$	$\begin{pmatrix} -s_{\beta_i} & 0 \\ 0 & 1 \\ c_{\beta_i} & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}$	$\begin{pmatrix} c_{\beta_i} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ s_{\beta_i} & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$	α_i β_i	$\begin{pmatrix} \dot{\alpha}_i \\ \dot{\beta}_i \end{pmatrix}$
6-DOF*	$\begin{pmatrix} \text{see} \\ \text{Table 2.1} \end{pmatrix}$	${}^0p_i$	$1_{6\times6}$		ε_i ${}^0p_i$	$\begin{pmatrix} \omega_i \\ v_i \end{pmatrix}$

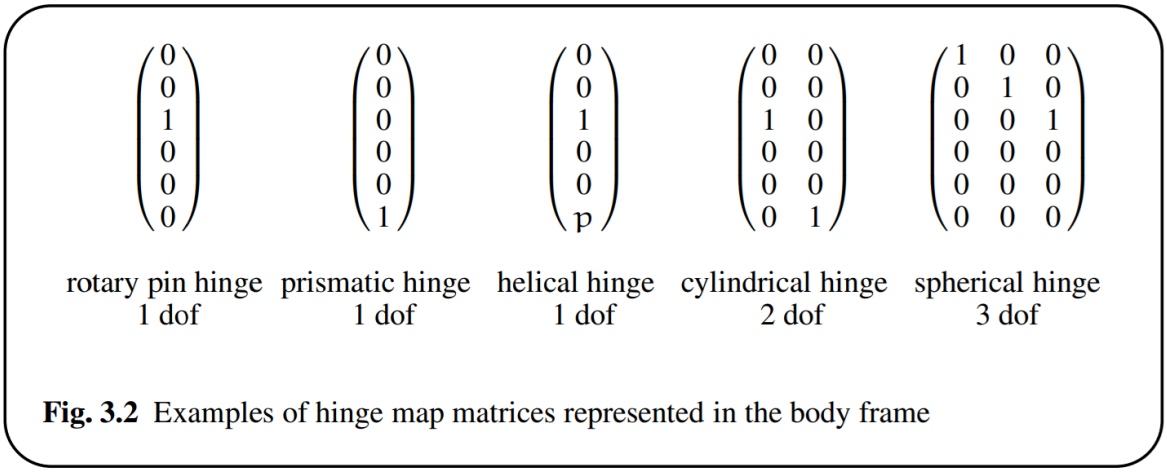


Fig. 3.2 Examples of hinge map matrices represented in the body frame

For a ball rolling on a surface, rolling (rolling means contact point has zero linear velocity) with no slipping can be treated as a hinge (integrable holonomic constraint). For an ellipsoid H is not constant. Joint map matrix not constant in ball frame, only in inertial frame.

Derivative formula for frames and joints

Since the motion of the bodies in a serial-chain are caused by hinge motions, we now look at formalizing the differential (i.e., velocity) kinematics for hinges. Towards this, differentiating ${}^{\mathbb{k}+\mathbb{1}}\mathbb{T}_{\mathbb{O}k}$ with respect to time yields:

$$\frac{d\,{}^{\mathbb{k}+\mathbb{1}}\mathbb{T}_{\mathbb{O}k}}{dt} = \begin{pmatrix} \tilde{\omega}\left(\mathbb{O}_k^+, \mathbb{O}_k\right) & v\left(\mathbb{O}_k^+, \mathbb{O}_k\right) \\ 0 & 0 \end{pmatrix}$$

When a hinge is simple with small number of degrees of freedom, it is a common practice to parametrize its velocity degrees of freedom using the time derivatives of the generalized coordinates. However, generalized velocity coordinates are more generally allowed to be *non-integrable quasi-velocities*. These can represent a more *convenient* choice for parametrizing the velocity degrees of freedom, and can decouple and simplify the dynamical and kinematical equations of motion. A classic example of the use of quasi-velocities is in the development of the dynamical equations of motion for a free rigid body. The use of the time derivatives of the orientation parameters (such as Euler angles, Euler parameters, etc.) as part of the 6-dimensional motion parameters *leads to complicated equations of motion*. However, the use of the *spatial velocity* vector as the set of the 6-dimensional generalized velocities leads to much *simpler* equations of motion [124]. The time derivatives of the generalized coordinates are related to the generalized velocities via a bidirectional, possibly *configuration dependent, map*. The relationship of the generalized velocities to the relative spatial velocity across the hinge is usually quite straightforward. The configuration dependent basis vectors for the hinge relative spatial velocity subspace are used as column vectors to define the configuration-dependent, full-rank hinge map matrix, 2 denoted H∗(k) ∈ R6×rv(k), for the kth hinge. H∗(k) defines a linear mapping of the generalized velocity coordinates vector, β(k) ∈ Rrv(k), for the hinge to the relative spatial velocity ΔV(k) across the hinge as follows:

$$\Delta_{\mathcal{V}}(k) = H^{\star}\beta(k)$$

In some instances, the above equation can have an additional bias term on the right of the above equation. The hinge constraint is said to be **catastatic** or **acatastatic** depending on whether this term is zero, or non-zero, respectively.
Differential form of holonomic constraints are also referred to as a Pfaffian form. When $\mathfrak{U} = 0$, the differential form of the constraints is said to be catastatic. When this is not the case, the constraints are said to be acatastatic. Clearly, scleronomic holonomic constraints lead to catastatic differential constraints, while rheonomic ones lead to acatastatic ones.

Spatial vector kinematic propagation

Using spatial vector notation propagation of velocities and acceleration 6-vectors are given respectively by

$$\mathcal{V}(k) = \Phi^{\star}(k+1,k)V(k+1) + \mathcal{H}^{\star}(k)\beta(k)$$

$$\mathcal{V}(k) = \Phi^{\star}(k+1,k)\mathcal{V}(k+1) + H^{\star}(k)\dot{\theta}$$

$$\alpha(k) = \Phi^{\star}(k+1,k)\alpha(k+1) + H(k)^{\star}\ddot{\theta}(k) + a(k)$$

where the velocity dependent **Coriolis spatial acceleration** $\mathfrak{a}(k) \in \mathcal{R}^{6}$ is defined as

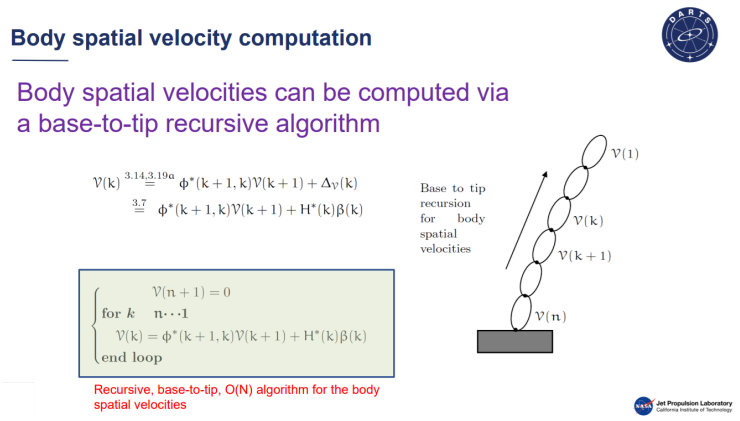
$$\mathfrak{a}(k) = -\Delta_{\mathcal{V}}^{\omega}(k)\mathcal{V}(k) + \frac{d\,\phi^{\star}(k+1,k)}{dt}\mathcal{V}(k+1) + \frac{d_{k+1}H^{\star}(k)}{dt}\dot{\theta}(k)$$

While $H^{\star}(k)$ is constant in the body frame for most hinge types, it can depend on the hinge generalized coordinates for more complex hinges. In this case, the time derivative of $H^{\star}(k)$ in equation is related to its gradient with respect to the generalized coordinate of the hinge as follows:

$$\frac{d_{k+1}H^{\star}(k)}{dt} = \left[\nabla_{\theta(k)}\left(H^{\star}(k)\right)\right]\dot{\theta}(k)$$

For the case when body-frame derivatives in equation are used, and $\mathbb{B}_k = \mathbb{O}_k$, the Coriolis acceleration, a(k), is given by the expression:

$$\mathfrak{a}(k) = \tilde{\mathcal{V}}(k)\Delta_{\mathcal{V}}(k) - \bar{\Delta}_{\mathcal{V}}(k)\Delta_{\mathcal{V}}(k) + \frac{d_{k+1}H^{\star}(k)}{dt}\dot{\theta}(k)$$



These expressions are at body level, since relates velocities between connected bodies pairs, connected by a joint.

Figure report an example of speeds propagation

Algorithm Recursive computation of link spatial velocities

$$\begin{cases} \mathcal{V}(n+1) = 0 \\ \textbf{for } k = n \cdots 1 \\ \quad \mathcal{V}(k) = \phi^*(k+1,k)\mathcal{V}(k+1) + H^*(k)\beta(k) \\ \textbf{end loop} \end{cases}$$

Use of generalized coordinates

Multibody state is the set of generalized coordinates and generalized velocities across all the hinges. Often generalized velocities are just generalized coordinate time derivatives (holonomic and integrable non holonomic coordinates/constraints). Quasi-velocities are a useful alternative, suitable both for holonomic and non holonomic case, corresponding to non integrable velocities coordinates. The term quasi-velocities designates pseudo-derivative since the velocity coordinates are not the time derivatives of any function of the generalized coordinates.

For a rigid body, an obvious choices of generalized velocity coordinates are the components of the spatial velocity, $\mathcal{V}(x)$, of a point x on the body represented either in the body frame, $\mathbb{B}$, or in the $\mathbb{I}$ inertial frame. For convenience, the point x is often chosen as the location of the body frame $\mathbb{B}$ itself. For the $\mathbb{B}$ frame representation, the generalized velocity coordinates are denoted $\beta_{\mathbb{B}} =^{\mathbb{B}} \mathcal{V}(x)$, while for the $\mathbb{I}$ frame representation, the generalized velocities are denoted $\beta_{\mathbb{I}} =^{\mathbb{I}} \mathcal{V}(x)$. The angular velocity components of the generalized velocities are examples of non-integrable velocity coordinates.

The definition of the *generalized forces* for a system depends on the choice of generalized velocities for the system. With β and $\mathcal{T}$ denoting the vectors of generalized velocities and forces, respectively for the system, the generalized forces must satisfy the requirement that

the $\mathcal{T}$ β expression represents the input *power*. Since the input power is independent of the specific choice of generalized velocities, this requirement implies that, if a new set of generalized velocities γ is chosen, the definition of the generalized forces must be changed accordingly. Thus, if γ is related to β by the relationship γ = A(θ)β, then it requires that $A^{-\star}(\theta)\,\mathcal{T}$ be the corresponding generalized forces. It is important to remember that the generalized forces *do not have* to directly correspond to *physical forces* on the system, and can be a transformation of these forces.

The *generalized acceleration* coordinates are defined as the time-derivatives of the generalized velocity coordinates. Different choices for the generalized velocity coordinates lead to different generalized acceleration coordinates.

Stacked vectors

We're talking so far of spatial vector and matrixes, whose properties are related to vector content and order specifically. A permutation of rows would change also mapping matrixes and equations shape. Another concept widely used wherever in mechanics, but having different meaning is one of stacked vectors, that's just a column vector of quantities whose order is strictly conventional and do not carry any physical meaning. Its use is predominantly notational, when we are interested in system level properties and stacking up component quantities into system level vectors. Stacking is applicable also to spatial vectors and quantities. This notational trick allows to pass from spatial vector for single corps, to system of corps equations, without changing much formalism (equivalent system-level expression).

sparse, only one subdiagonal with inter-body rigid transformation matrix elements
Rows: parent body Columns: child body

Operator for relative hinge spatial velocities
Block-diagonal with joint map matrix elements and non-square. The block-diagonal nonzero entries are the transpose of the configuration dependent joint map matrices for the body hinges
Every power of results in a matrix with the sub-diagonal shifted one step lower. At the nth power, the result is zero. Hence is nilpotent!

spatial velocity on the outboard side of the hinge

Nilpotent matrix inversion

A square matrix is said to be nilpotent if one of its powers becomes 0, i.e. if for some n

$$U^n = 0$$

For a nilpotent , we have (series expansion)

$$(I+U)^{-1} = I + U + U^2 + U^3 + \cdots + U^{n-1}$$

Series expansion terminates after only a finite number of terms for nilpotent matrix, hence the 1-resolvent inverse is well defined. Define:

$$W = I + U + U^2 + U^3 + \cdots + U^{n-1}$$

thus:

$$UW = WU = U + U^2 + U^3 + \cdots + \underbrace{U^n}_{=0} = W - I$$

$$I = W - UW = W - WU = W\left(I - U\right) = \left(I - U\right)W$$

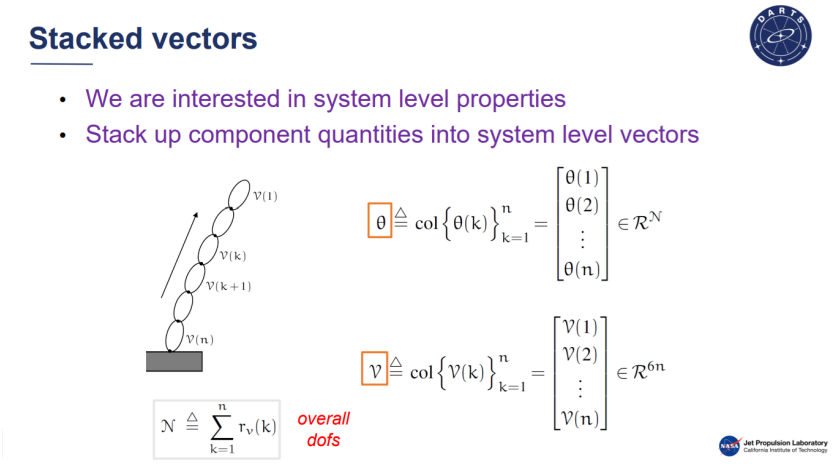
$$W = \left(I - U\right)^{-1}$$

Hinge coordinates staked vectors

Now we introduce stacked vectors required to define system level relationships. We begin by defining the stacked θ as stacked vector consisting of the component body-level θ(k) generalized coordinates assembled into a single large vector.

The number of overall velocity degrees of freedom for the system is denoted $\mathcal{N}$, and is defined as the sum of all the individual hinge velocity degrees of freedom, i.e.,

$$\mathcal{N} = \sum_{k=1}^{N} r_v\left(k\right)$$



System level propagation of kinematics

Taking body level propagation of kinematics and using system level stacked notation, one obtain a multibody system level kinematic propagation system of equations.

$$\mathcal{V} = \mathcal{E}_{\phi}^{\star}\mathcal{V} + H^{\star}\dot{\theta}$$

$$\alpha = \mathcal{E}_\phi^\star \alpha + H^\star \ddot{\theta} + \mathfrak{a}$$
$$\mathfrak{f} = \mathcal{E}_\phi \mathfrak{f} + M\alpha + \mathfrak{b}$$
$$\mathfrak{T} = H\mathfrak{f}$$

equivalent system-level implicit operator expressions. However one observe $\mathcal{E}_\phi$ is nilpotent, so one can use analytical inversion formula. Use identity to convert implicit operator expressions into explicit ones

$$\mathcal{V} = \Phi^\star H^\star \dot{\theta}$$
$$\alpha = \Phi^\star \left(H^\star \ddot{\theta} + \mathfrak{a} \right)$$
$$\mathfrak{f} = \Phi \left(M\alpha + \mathfrak{b} \right)$$
$$\mathfrak{T} = H\mathfrak{f}$$

Combine the operator expressions to obtain the system level equations of motion

$$\mathfrak{T} = H\Phi \left(M\Phi^\star \left(H^\star \ddot{\theta} + \mathfrak{a} \right) + \mathfrak{b} \right) = M\left(\theta\right)\ddot{\theta} + C\left(\theta, \theta\right)$$

These equations of motion hold for any tree/branched system. The equations of motion derived using the Newton-Euler approach are the same as we would have obtained using alternative approaches such as the Lagrangian approach:

- The mass matrix term equivalence is easy to see
- The Coriolis term takes a lot more work, but can be shown to be equivalent

What obtained allow us to write down Coriolis terms in a different than usual way, not involving any derivative of inertia matrix, similarly to RNE algorithm.

$$\mathfrak{C}\left(\theta, \dot{\theta}\right) = H\Phi \left(M\Phi^\star \mathfrak{a} + \mathfrak{b} \right)$$

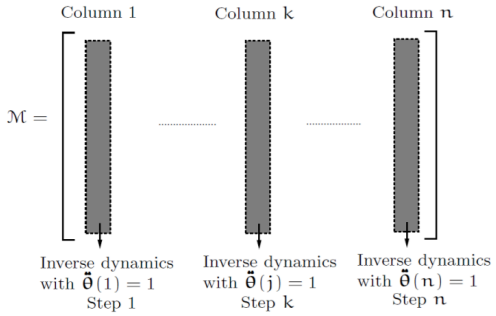
Update the force balance equation to include external forces, Using stacked notation
The equations of motion thus take the form

$$\mathcal{M}\left(\theta\right)\ddot{\theta} + \mathfrak{C}\left(\theta, \dot{\theta}\right) = \mathcal{T}$$

The Coriolis vector is zero when velocities, external forces and gravity are zero. Inverse dynamics with all-zero generalized accel except kth element being 1 yields the kth column of the mass matrix as usual. Repeating this procedure for each element of the generalized accels vector to get the full mass matrix. Algorithm already presented elsewhere, is depicted in figure. The CRB-based algorithm is however faster. CRBs based decomposition of the mass matrix is:

Structure of the algorithm

Algorithm consists of a sequence of inverse dynamics computations



$$M = H\mathcal{R}H^\star + H\tilde{\Phi}\mathcal{R}H^\star + H\mathcal{R}\tilde{\Phi}^\star H^\star$$

$$\Phi M \Phi^\star = \mathcal{R} + \tilde{\Phi}\mathcal{R} + \mathcal{R}\tilde{\Phi}^\star$$

$$y = \Phi\left(\mathfrak{b} + \mathcal{E}_\phi \mathcal{R}\left(H^\star \ddot{\theta} + \mathfrak{a}\right)\right)$$

Relation with

$\Phi^\star H^\star$ matrix elements are the “partial velocities” from Kane’s method.

We would neither get the CRB decomposition of the mass matrix or the recursive NE inverse dynamics structure if we evaluated into a partial velocities matrix! • The operator form is key to preserving structure.

Kinematics propagation using stacked vectors

While we have operation expression for the system level spatial velocities, it is implicit! How to get rid of this to get an explicit

Strictly lower triangular, square, singular and nilpotent. Only the first sub-diagonal has nonzero elements. The non-zero entries are the configuration dependent 6x6 inter-link rigid body transformation matrices (configuration dependent)
is nilpotent for a tree system, and we can thus define its 1-resolvent

Kinematics chains

Essentially the use of kinematics chains pose conceptual basis for treatment of generic systems of connected bodies (even flexible), included closed chains. It's just matter of defining how to connect frames.
Forward kinematics problem is to compute the pose of any body in the system. The computation of body poses can be done recursively from base to tip (or vice versa)

The relative pose of any pair of bodies j and k can also be computed recursively:

$$ {}^{j}\mathbb{T}_{k} = {}^{j}\mathbb{T}_{j-1}{}^{j-1}\mathbb{T}_{j-2}\dots{}^{k+1}\mathbb{T}_{k} $$

Exemplificative algorithm is reported in figure

Algorithm Recursive computation of link transformations

$$ \begin{cases} {}^{\mathbb{I}}\mathbb{T}_{n+1} = \mathbf{I} \\ \textbf{for } k = n \cdots 1 \\ \quad {}^{\mathbb{I}}\mathbb{T}_{k} = {}^{\mathbb{I}}\mathbb{T}_{k+1} \cdot {}^{k+1}\mathbb{T}_{k} \\ \textbf{end loop} \end{cases} $$

Jacobian operator

There are typically points of interest on bodies that we will refer to a nodes, eg.

- End-effector frame for a robot
- Attachment points for actuators and sensors
- Reference frames for control algorithms

The spatial velocity for a node can be obtained from that of its parent body as follows:

$$ \mathcal{V}\left(\mathbb{O}_{k}^{i}\right) = \Phi^{\star}\left(k, \mathbb{O}_{k}^{i}\right)\mathcal{V}\left(k\right) $$

There are times when we need to narrow attention to the nodes. Then we introduce a Pick-off operator for selecting specific points of interest.

$$ \mathcal{V}_{nd} = \mathcal{B}^{\star}\mathcal{V} $$

defined as

$$ \mathcal{B} = \begin{pmatrix} \Phi\left(1, O_{1}^{0}\right) \\ \vdots \\ 0 \\ 0 \end{pmatrix} $$

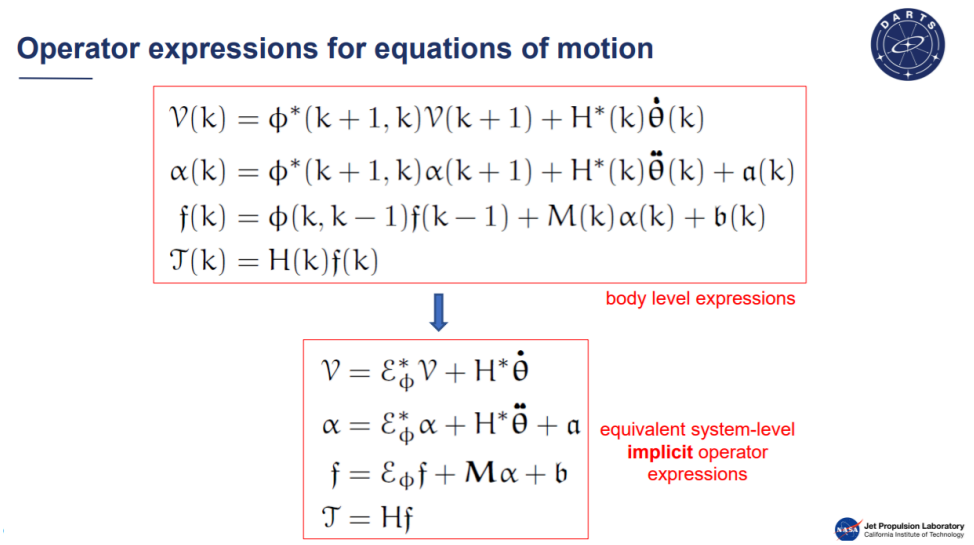
$$ \mathcal{V} = \Phi^{\star}H^{\star}\dot{\theta} $$

Jacobian definition results to be in SOA, from definition:

$$\mathcal{V} = \mathcal{J}\,\dot{\theta}$$

$$\mathcal{J} = \mathcal{B}^{\star}\Phi^{\star}H^{\star}$$

The Jacobian relates the generalized velocities to the spatial velocity of a node of interest



Euler equations

Euler equations in spatial vector notation are straightforward. Equations capture both linear and rotational dynamics and their coupling; Used several choices for generalized velocities possible; Structure remained the same, variation in gyroscopic term

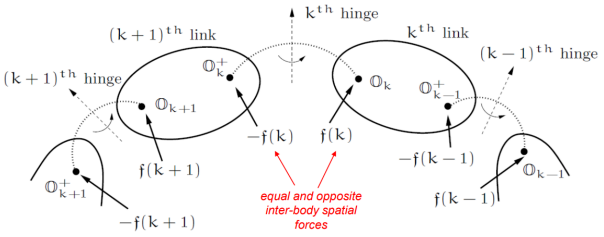
$$\mathfrak{f}(z) = M(z)\dot{\beta}_{\mathcal{J}}(z) + \mathfrak{b}_{\mathcal{J}}(z)$$

Spatial vector equations for dynamics: Newton Euler algorithm

Kinematic propagation is obtained using above equations.
In spatial vector notation forces and torques for each rigid body are collected in one single 6 vector. consequently forces and torques transmission from one body to another is written easily:

$$f(k) = \Phi(k,k-1)f(k-1) + M(k)\alpha(k) + b(k)$$

Force balance for a single link



$$f(k) - \phi(k, k-1) f(k-1) = M(k)\alpha(k) + b(k)$$

overall spatial forces from the child and parent bodies

Jet Propulsion Laboratory California Institute of Technology

where Φ play the role of a rotation matrix between body frames. Once forces and torques vectors acting on body k are calculates, we can project them on joint subspace using hinge map to get joint force

$$\tau(k) = H(k) f(k)$$

Resulting overall NE inverse dynamics algorithm is depicted in figure:

Algorithm Newton–Euler inverse dynamics algorithm

$$\begin{cases} \mathcal{V}(n+1) = \mathbf{0}, \quad \alpha(n+1) = \mathbf{0} \\ \textbf{for } k = n \cdots 1 \\ \quad \mathcal{V}(k) = \phi^*(k+1,k)\mathcal{V}(k+1) + H^*(k)\dot{\theta}(k) \\ \quad \alpha(k) = \phi^*(k+1,k)\alpha(k+1) + H^*(k)\ddot{\theta}(k) + a(k) \\ \textbf{end loop} \end{cases}$$

$$\begin{cases} f(0) = \mathbf{0} \\ \textbf{for } k = 1 \cdots n \\ \quad f(k) = \phi(k,k-1) f(k-1) + M(k)\alpha(k) + b(k) \\ \quad \mathcal{T}(k) = H(k) f(k) \\ \textbf{end loop} \end{cases}$$

External forces

Our development of the equations of motion so far has assumed that the only forces on the links are from the interaction forces among the links and has not taken into account any external forces on the system. Thus, the free-body equations of motion for the kth link in (5.1) only involve the f(k) and f(k−1) inter-link forces. We now look at extensions to handle external forces on the system. Let us assume that external spatial forces are being applied at nodes on the bodies in the serial-chain. Adopting the notation for nodes from Sect. 3.6 on page 53, Oi k denotes the ith such node on the kth link. Let f i ext(k) denote the external spatial force being applied at the Oi k node. This spatial force is effectively φ(k,Oi k)fi ext(k)

at the Bk body frame. Summing up all such external spatial forces on the kth link, the free-body equations of motion for the link in (5.1) are modified as follows:

Including external forces on bodies

Update the force balance equation to include external forces

$$\mathfrak{f}(k) - \phi(k, k-1)\mathfrak{f}(k-1) + \boxed{\sum_i \phi(\mathbb{B}_k, \mathbb{O}_k^i)\mathfrak{f}_{ext}^i(k)} = M(k)\alpha(k) + \mathfrak{b}(k)$$

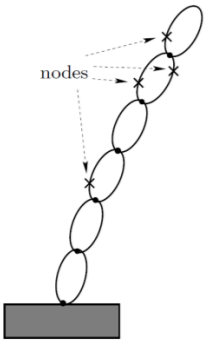
external forces

Using stacked notation

$$\mathfrak{f}_{ext} = \text{col}\left\{\mathfrak{f}_{ext}^i(k)\right\} \quad \in \mathcal{R}^{6n_{nd}}$$

$$\mathfrak{f} = \mathcal{E}_\phi \mathfrak{f} - \boxed{\mathcal{B}\mathfrak{f}_{ext}} + \mathbf{M}\alpha + \mathfrak{b}$$

$$\mathfrak{f} = \phi\left(\mathbf{M}\alpha + \mathfrak{b} - \boxed{\mathcal{B}\mathfrak{f}_{ext}}\right)$$



Jet Propulsion Laboratory
California Institute of Technology

Transforming SOA operator expressions into recursive algorithms

SOA analysis that exploits mathematical structure of dynamics

*Mapping to **structure based**, fast recursive algorithms*

Dynamics properties	→	Transformed Expressions	→	Low-order structure-based algorithms
• *General approach*		• *Exploit structure*		• *Faster*
• *Concise*		• *Get new insights*		• *More robust*
• *Rich vocabulary*		• *Solve new problems*		

"Structure-based": Because the pattern of the recursive algorithms is entirely driven by the underlying multibody topology.

Jet Propulsion Laboratory
California Institute of Technology

Composite Rigid Body Inertia matrix

Using SOA is easy to derive a compact form iterative algorithm for CRBI matrix. This is an iterative process as shown by equation (**Lyapunov equation**):

$$\mathcal{R}(k) = \Phi(k, k-1)\mathcal{R}(k-1)\Phi^\star(k, k-1) + M(k)$$

This recursive equation is referred to as a Lyapunov equation in the optimal estimation literature.
Or in algorithmic form:

Algorithm 4.1 Recursive computation of Composite Body Inertias

$$
\begin{cases}
\mathcal{R}(0) = \mathbf{0} \\
\textbf{for } k = 1 \cdots n \\
\quad \mathcal{R}(k) = \phi(k, k-1)\mathcal{R}(k-1)\phi^*(k, k-1) + M(k) \\
\textbf{end loop}
\end{cases}
$$

This recursive procedure is also known as a ***Lyapunov recursion*** since it is closely associated with the Lyapunov equation for composite body inertias. A depiction of possible algorithm for CRBI is given in figure:

Algorithm Recursive computation of the mass matrix

$$
\begin{cases}
\mathcal{R}(0) = \mathbf{0} \\
\textbf{for } k = 1 \cdots n \\
\quad \mathcal{R}(k) = \phi(k, k-1)\mathcal{R}(k-1)\phi^*(k, k-1) + M(k) \\
\quad \begin{cases}
X(k) = \mathcal{R}(k)H^*(k), \quad \mathcal{M}(k, k) = H(k)X(k) \\
\textbf{for } j = (k+1) \cdots n \\
\quad X(j) = \phi(j, j-1)X(j-1) \\
\quad \mathcal{M}(j, k) = \mathcal{M}^*(k, j) = H(j)X(j) \\
\textbf{end loop}
\end{cases} \\
\textbf{end loop}
\end{cases}
$$

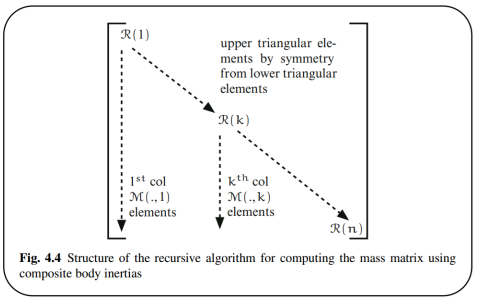


Fig. 4.4 Structure of the recursive algorithm for computing the mass matrix using composite body inertias

Lyapunov recursion and parallel with estimation theory

Here we describes the generalization of this composite body inertia algorithm for tree-topology systems.

Some of the terminology we have used thus far has its roots in optimal estimation theory. We take a look at such parallels in this section because the mathematical connections run deep. Let us begin by considering a discrete-time, time-varying, linear dynamical system described by the following state-transition equations:

$$
\begin{cases}
x(k) = \Phi(k, k-1)x(k) + w(k) \\
\mathcal{T}(k) = H(k)x(k)
\end{cases}
$$

x(k) = φ(k,k−1)x(k−1)+ w(k)
T(k) = H(k)x(k) (6.45)

In the above equations, we have deliberately borrowed notation from the multibody dynamics context to this time-domain problem, to highlight the mathematical parallels across the domains. However, the meaning of the symbols are quite different and are as follows: k represents the time variable, x(k) is the system state vector at time k, w(k) is an input white noise process with covariance M(k) • φ(k,k−1) is a time-varying state-transition matrix that propagates the state at time (k−1) to the state at time k

Process Covariances

In the optimal estimation literature, it is known that the covariance matrix for the x(k) system state, R(k), satisfies Lyapunov equation. This recursive equation comes from the optimal estimation literature. Equation is identical in form to the recursive relationship for composite

body inertias. The cross-covariances of the system state at different time instants are given by the elements of the φMφ* matrix, where the φ, H, and M matrices have structure identical to the matrix definitions in (3.32), (3.37), and (4.4), respectively. The cross-covariances of the output process, T(k), are the elements of the M = HφMφ*H* matrix, which is identical in form to the mass matrix of the serial-chain system.

Optimal filtering

The *optimal filtering* problem consists of generating an optimal estimate of the unknown internal state, x(k), from just the knowledge of the current and past noisy observations, namely the T(j) (joint actuators torques and forces) values prior to, and including, time k. The *optimal smoothing* problem consists of using additional observations after time k to optimally improve the filtered state estimate at time k. Thus, the optimal filtering process is said to be *causal*, since it is only uses current and past observations. On the other hand, the optimal smoothing process is *anti-causal,* since it is allowed to use future observations to improve the current estimate. Kalman and others developed elegant solutions to the optimal filtering and smoothing problems. The optimal filter estimate of the system state at time k, denoted $z^+(k)$, satisfies a recursive relationship with the optimal filter estimate, $z^+(k-1)$, from the previous time step k − 1. We sketch out this *two stage filter estimate generation technique* that takes place at each time step:

1. First, with no additional observations, the optimal filter estimate from the previous time k−1,$z^+(k-1)$, is propagated to form a prediction $z(k)$ of the optimal filter estimate of x(k) at time k.
2. Next, the new observation T(k) at time k, is processed to remove the part that is predictable from the past observations, to create a residual innovations term, (k), containing just the new information. This innovation term is used to update equation.

the predicted optimal filter estimate, z(k), to obtain the optimal filter estimate, $z^+(k)$, at time k. This 2-step recursive procedure is applied at each new time step as new observation values are obtained. The covariance of the optimal filter estimate error, x(k)− z(k), at time k, denoted P(k), satisfies the equation

P(k) = ψ(k,k−1)P(k−1)ψ*(k,k−1) +M(k) (6.47)

This equation is referred to as the *discrete form of the Riccati equation*. This equation is identical in form to the for articulated body inertia quantities. The definition of ψ(k,k−1) is mathematically identical to that in (6.29). Given the optimal filtered estimate at time (k−1), $z^+(k-1)$, the best prediction of the system state (without any additional observations), z(k), is obtained by simply propagating this estimate using the state-transition matrix as follows:

z(k) = φ(k,k−1)z+(k−1) (6.48)

Once the new observation at time k, T(k), is available, it can be used to first extract the new information in the observation, (k), and can then be used to update z(k) to obtain the new optimal filtered estimate at time k, $z^+(k)$, as follows:

(k) = T(k) −H∗(k)z(k)
z
+(k) = z(k) +G(k)(k) (6.49)
D(k) = H(k)P(k)H∗(k) is the covariance of the (k) innovations process. G(k) =
P(k)H∗(k)D−1(k)
is known as the **Kalman gain** in the estimation literature, and depends directly on the P(k) covariance obtained as a solution to the Riccati equation. Observe that the optimal filter estimate at time k only depends on the past observations and, hence, this optimal filter is said to be a causal filter. Equations are identical in form, with those from the multibody context. Thus, we have observed that several quantities in the articulated body model are mathematically equivalent to terms in Kalman optimal filtering for discrete-time systems.

Optimal Smoothing The parallels continue

.The optimal smoothed estimate, of the x(k) state at time k, f(k), is based on the knowledge of the T(j) observations over the full time interval 1 through n, and not on just the past observations, as for the $z^{+}(k)$ optimal filtered estimate. The smoothed estimate, f(k), can be obtained by updating the filtered estimate as follows:
f(k) = z(k) +P(k)α(k)=z+(k) +P+(k)α+(k)

Articulated Body Model for Serial Chains The α and α+(k) quantities above are obtained by an anti-causal recursion that uses the future observations in a time-reversed recursion that is mathematically identical to (6.43). Equation (6.50) is precisely the form of the force decomposition from the articulated body model in (6.32) and (6.34)!

Extensions

The mathematical parallels between estimation theory and multibody dynamics are extensive. In the factorization of the mass matrix and an analytical expression for its inverse results mirror *spectral factorization* techniques from estimation theory. These parallels were first identified by Rodriguez and discussed in [139–143, 145]. However, it is also appropriate to point out areas where the parallels start to **breakdown** and the domains diverge. Unlike the strict sequential order imposed by time for discrete-time systems, *no such ordering* is required for the links in multibody systems. The link numbering is not restricted to the parent/child sequence. Moreover, when we explore non-serial-chain multibody systems with branched topologies, the sequential parallels with time are again *less obvious*. However, as we will see in later chapters, it is remarkable that, with modest extensions, several of the insights developed by exploiting the mathematical parallels between time domain and serial-chain system domains continue to hold for more general multibody systems.
These sections considerations comes from [Jain: Robot and multibody dynamics].

Inertia matrix factorization and inverse

Inertia matrix using spatial vector operator algebra allows to express efficiently its structure leading to illustrative factorization based only on hinge map, and

$$M = (I + H\Phi K)D(I + H\Phi K)^{\star}$$

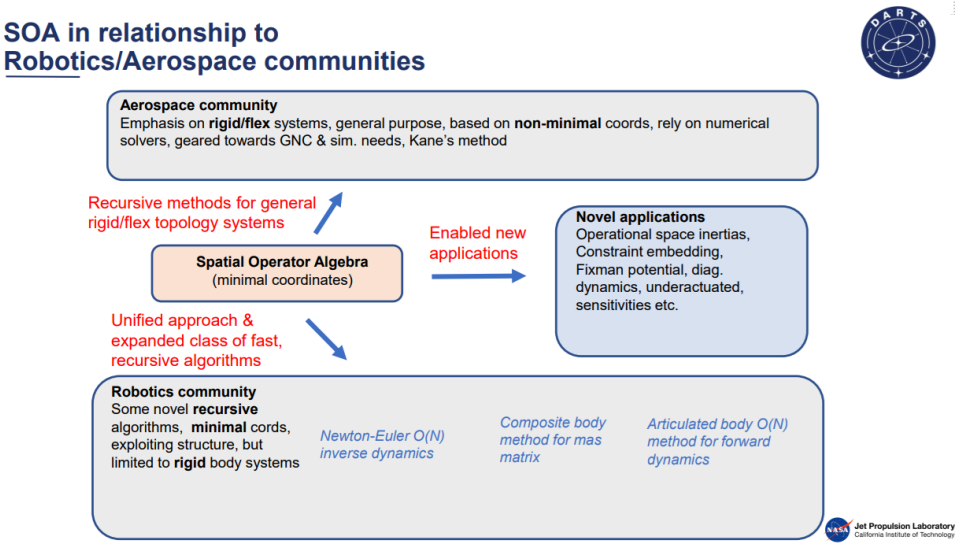
whose structure shows role and similarities with kalman filtering components, giving inertia matrix as a spatial process. Tis notation drive us to a simpler form of inertia matrix inverse, using analytical factor inverse (valid for nilpotent matrixes Nilpotent matrix):

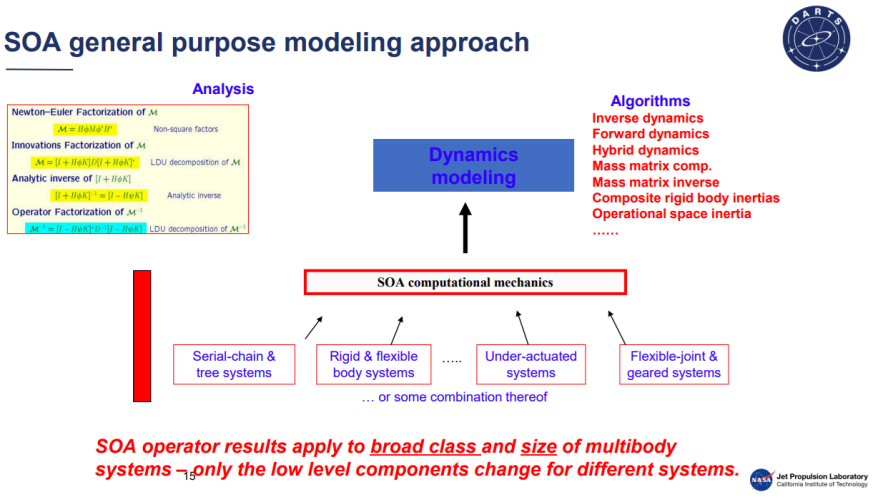
$$(I + H\Phi K)^{-1} = (I - H\Phi K)$$

Indeed analytical inversion of a matrix is a considerably complex task, even with powerful computer and with minimum internal matrix complexity, then analytical forms of inverse, where available,are welcome. This is indeed Kalman matrix inverse lemma used here and in kalman filtering theory.

$$M^{-1} = (I + H\Phi K)^{-\star}D^{-1}(I + H\Phi K)^{-1}$$

$$M^{-1} = (I - H\Phi K)^{\star}D^{-1}\,(I - H\Phi K)$$





SOA operator results apply to broad class and size of multibody systems – only the low level components change for different systems.

Fixman potential and sensitivity analysis

Sensitivities of $[I - H\phi\mathcal{K}]$ and $[I + H\phi\mathcal{K}]$ show that

$$[I + H\phi\mathcal{K}]_{\theta_i} = H\phi\left[\tilde{\mathcal{H}}^{\omega}_{=i}\tilde{\phi}\mathcal{P} + \mathcal{E}_{\phi}\tilde{\mathcal{H}}^{v}_{=i}\phi\mathcal{P} + \bar{\tau}\breve{\lambda}_{\theta_i}\right]H^{\star}\mathcal{D}^{-1}$$

$$[I - H\psi\mathcal{K}]_{\theta_i} = -H\psi\left[\tilde{\mathcal{H}}^{\omega}_{=i}\phi\mathcal{K} + \mathcal{E}_{\phi}\tilde{\mathcal{H}}^{v}_{=i}\phi\mathcal{G} + \bar{\tau}\breve{\lambda}_{\theta_i}H^{\star}\mathcal{D}^{-1}\right](I - H\psi\mathcal{K})$$

The following developed expression for $\log\{\det\{\mathcal{M}\}\}$ and its partial derivatives with respect to generalized coordinates. This term appears in expressions for the Fixman potential [57,72] in the context of internal coordinate molecular dynamics [88]. The Fixman potential is used in internal coordinate molecular dynamics simulations, to correct for systematic biases introduced in the computation of statistical thermodynamic quantities. Its partial derivatives with respect to the generalized coordinates are the generalized forces resulting from the Fixman potential.

$$\frac{\partial \log\{\det\{\mathcal{M}\}\}}{\partial\theta_i} = 2\,Trace\left\{\mathcal{P}(i)\gamma(i)\tilde{\mathcal{H}}^{\star}(i)\right\}$$

Application of mass matrix determinant: Fixman Potential

The Fixman potential is needed in molecular dynamics simulations for correcting statistical biases

$$\mathrm{U_f} \triangleq \log\{\det\{\mathcal{M}\}\} \qquad \prod_{k=1}^{n}\det\{\mathcal{D}(k)\}$$

Computing and using it has been an intractable problem for decades

$$\frac{\partial \log\{\det\{\mathcal{M}\}\}}{\partial\theta_i} = 2\,\mathrm{Trace}\left\{\mathcal{P}(i)\Upsilon(i)\tilde{H}^{*}(i)\right\}$$

Torque from the Fixman potential

Available from standard ATBI computations

*Explicit simple expression via SOA for longstanding **intractable** problem.*

Jet Propulsion Laboratory
California Institute of Technology

Use of SOA for diagonalizing dynamics

Classic form of dynamics equations arising from natural adoption of joints variable universally used in robotics and in many other multibody mechanical systems are characterized by high equations coupling, undesirable because leading to complex dynamics and stiffing numerical simulations as well as analytical study of system, reducing numerical stability, accuracy, robustness, predictability and reliability of results.

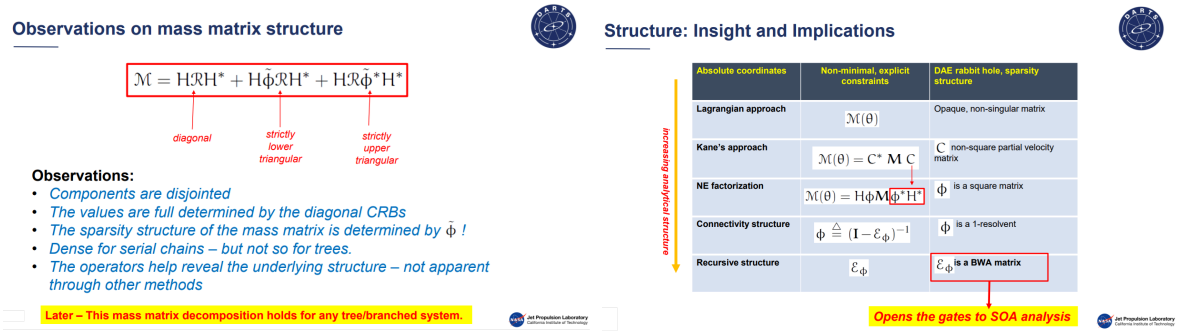
Globalization of diagonalization transformations requires vanishing of curvature tensor associated to inertia matrix. This is rarely the case.
The use of diagonalizing coordinates

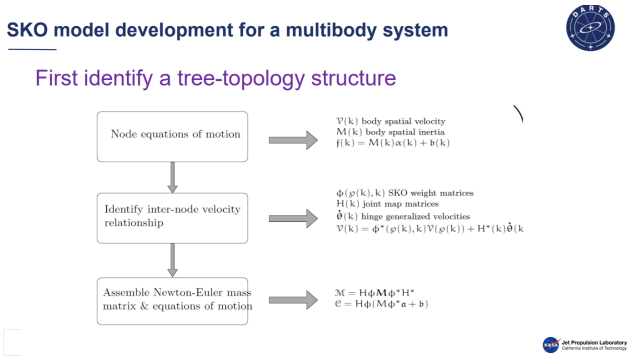
$$M(\theta)\ddot{\theta} + C\left(\theta, \dot{\theta}\right) = T \Rightarrow \dot{\nu} + C(\theta, \nu) = \epsilon$$

$$\underbrace{M(\theta)\ddot{\theta} + C\left(\theta, \dot{\theta}\right) = T}_{\text{highly coupled}} \Rightarrow \underbrace{\dot{\nu} + C(\theta, \nu) = \epsilon}_{\text{largely decoupled}}$$

$$\dot{\theta} \rightarrow \nu = M^{1/2}\dot{\theta}$$

- These smooth diagonalizing transformation of coordinates always exist
- These are non integrable time derivatives of quasi coordinates
- Can derive closed form operator expression and computational algorithms for $C(\theta, \upsilon)$
- There are analogies with rigid body equations (non working part)
- Simpler control law are possible
- Reduce complexity and burden for numerical integration, simulation
- Reduce complexity and burden for statistical mechanics





Comparison of composite and articulated body models

Variable	Composite Body	Articulated Body
Relative hinge acceleration	$\ddot{\theta}$	ν
α	$\phi^* H^* \ddot{\theta}$	$\psi^* H^* \nu$
$\alpha(k)$	$\phi^*(k+1,k)\alpha(k+1) + H^*(k)\ddot{\theta}(k)$	$\psi^*(k+1,k)\alpha(k+1) + H^*(k)\nu(k)$
α with respect to α^+	$\alpha^+ + H^*\ddot{\theta}$	$\overline{\tau}^*\alpha^+ + H^*\nu$
		$(\overline{\tau}^*\alpha = \overline{\tau}^*\alpha^+)$
ν	$\mathcal{G}^*\alpha^+ + \ddot{\theta}$	$\mathcal{G}^*\alpha$
Effective inertia	$\mathcal{R}$	$\mathcal{P}$
Relationship to $\mathbf{M}$	$\mathcal{R} - \mathcal{E}_\phi \mathcal{R} \mathcal{E}_\phi^*$	$\mathcal{P} - \mathcal{E}_\psi \mathcal{P} \mathcal{E}_\psi^*$
Inertia recursions	$\mathcal{R}^+(k) = \mathcal{R}(k)$	$\mathcal{P}^+(k+1) = \overline{\tau}(k)\mathcal{P}(k)\overline{\tau}^*(k)$
	$\mathcal{R}(k+1) = \phi(k+1,k)\mathcal{R}^+(k)\phi^*(k+1,k) + M(k)$	$\mathcal{P}(k+1) = \phi(k+1,k)\mathcal{P}^+(k)\phi^*(k+1,k) + M(k)$
f on ($-$ side)	$\mathcal{R}\alpha + \gamma$	$\mathcal{P}\alpha + \zeta$
Correction force ($-$ side)	$\gamma = \tilde{\phi}\mathcal{R}H^*\ddot{\theta}$	$\zeta = \tilde{\phi}\mathcal{P}H^*\nu$
f on $+$ side	$\mathcal{R}^+\alpha^+ + \gamma^+$	$\mathcal{P}^+\alpha^+ + \zeta^+$
Correction force ($+$ side)	$\gamma^+ = \gamma + \mathcal{R}H^*\ddot{\theta}$	$\zeta^+ = \zeta + \mathcal{P}H^*\nu$

Jet Propulsion Laboratory
California Institute of Technology

Table 2.2 A comparison of the properties of the different formulations for the dynamics of a single rigid body

Formulation	$\mathbb{I}$ deriv, $\mathbb{C}$	$\mathbb{I}$ deriv, z	$\mathbb{B}$ deriv, z	Inertial ref, $\mathbb{I}$
Section	2.3.1	2.3.2	2.4	2.5
Gen. vel. β	${}^{\mathbb{I}}\mathcal{V}(\mathbb{C})$	${}^{\mathbb{I}}\mathcal{V}(z)$	${}^{\mathbb{B}}\mathcal{V}(z)$	$\mathcal{V}_{\mathbb{I}}$
Gyroscopic force $\mathfrak{b}$	$\overline{\mathcal{V}}^{\omega}(\mathbb{C})\mathcal{M}(\mathbb{C})\mathcal{V}^{\omega}(\mathbb{C})$	$\overline{\mathcal{V}}^{\omega}(z)\mathcal{M}(z)\mathcal{V}^{\omega}(z)$	$\overline{\mathcal{V}}(z)\mathfrak{h}(z)$	$\overline{\mathcal{V}}_{\mathbb{I}}\mathfrak{h}_{\mathbb{I}}$
Conserved spatial momentum	$\surd$			$\surd$
Non-working $\mathfrak{b}$	$\surd$		$\surd$	$\surd$
Independent of linear velocity	$\surd$	$\surd$		

Observations

Motion invariances for a rigid body. Examined the conservation properties of the kinetic energy and spatial angular momentum of a body in the absence of external forces. These conservation properties are a consequence of *Noether's theorem* which shows that certain symmetries of the system Lagrangian of a system result in integrals of motion, i.e., quantities that are conserved with time. For rigid body example, the non-dependency of the kinetic energy of the body on time results in the conservation of kinetic energy over time. Moreover, the non dependence of the kinetic energy on the location as well as the orientation of the rigid body results in the *conservation of the linear and angular momentum* properties for the body

Remark 2.1 Gyroscopic force b(z) does no work. We show now that the gyroscopic spatial force b(z) in the equations of motion in Lemma 2.3 does no work. The power generated by the gyroscopic force b(z) is

$$\mathcal{V}^{\star}(z)\mathfrak{b}(z) = \mathcal{V}^{\star}(z)\bar{\mathcal{V}}(z)\mathfrak{h}(z) = 0$$

V∗(z)b(z) 2.28 = V∗(z)V(z)h(z) 1.24 = 0 (2.29)

confirming that b(z) is a non-working force.

Remark 2.2 Conservation of kinetic energy for a rigid body. We now show that the equations of motion in Lemma 2.3 further imply that the kinetic energy of a body is conserved in the absence of external forces, i.e., f(z) ≡ 0. The time derivative of the kinetic energy is given by

1 2 dV∗(z)M(z)V(z) dt = V∗(z)M(z)β˙ B(z) 2.28 = −V∗(z)b(z) 2.29 = 0

$$\frac{1}{2}\frac{\mathrm{d}}{\mathrm{d}t}\left(\mathcal{V}^{\star}(z)M(z)\mathcal{V}(z)\right) = \mathcal{V}^{\star}(z)M(z)\dot{\beta}_{\mathbb{B}}(z) = -\mathcal{V}^{\star}(z)\mathfrak{b}(z) = 0$$

This implies that the kinetic energy of a body is conserved when the external forces on the body are zero.

Remark 2.3 Body dynamics invariance across Newtonian frames: Newtonian frames are free-falling frames. A frame moving at fixed linear velocity with respect to a Newtonian frame is also a Newtonian frame. Frames that are accelerating or rotating with respect to a Newtonian frame are not Newtonian frames. In general, the dynamics of a multibody system are invariant across Newtonian frames but not across non-Newtonian ones. We examine the implications of this invariance property for the single rigid body system that we have studied so far. We should expect that formulations using derivatives with respect to Newtonian frames (e.g. inertial frame derivatives) would have equations of motion that are independent of the linear velocity of the body. On the other hand we can expect that formulations using derivatives in the non-Newtonian frames (e.g. rotating body frames) would not necessarily retain this invariance property. These conjectures can be verified by examining the expression for the bI(z) spatial force, for the inertial frame derivative dynamics, and noting that it is independent of the linear velocity. On the other hand, the corresponding gyroscopic spatial force, b(z) for body frame derivatives does depend on the linear velocity of the body.

Remark 4.2 The angular velocity as quasi-velocities. While the equations of motion based on *quasi-velocities* appear more complex than those in the rigid body rotational equations of motion are a good example where the converse is true. Focusing on just the rotational dynamics of a rigid body, we choose its angular velocity, ω, *as the generalized velocities* for the body. For this case, the Lagrangian L´ = 1 2ω∗J ω is *independent* of the θ generalized coordinates. While ω is *nonintegrable*, there is nevertheless a smooth and invertible transformation A(θ) such that I ω = A(θ)θ˙. Equation providing an explicit expression for A(θ) for one such choice of Euler angles for θ. It can be shown that A−∗γ∗ = ω. Since J is constant in the body frame, (4.44) reduces to the following simpler form of the equations of motion: J ω˙ + ωJ ω = A−∗f (4.47) In contrast, if we usually can and prefer to choose to work with *integrable* generalized velocities, such as the time derivatives of Euler angle

generalized coordinates, the resulting equations of motion are significantly more complex but allows for connecting position and speed by integral equations.

Remark 5.1 a(k) for simple hinges. While (5.11) provides the general expression for the link Coriolis acceleration spatial vector, it is usually simpler in practice. When the joint map matrix H∗(k) is *constant* in the body frame, the last time derivative in (5.11) vanishes. Also, if the hinge is either purely prismatic or purely rotational then the second term vanishes as well. For these cases, a(k) simplifies to
a(k) = V(k)ΔV(k) (5.15)

$$\mathfrak{a}\left(k\right)=\tilde{\mathcal{V}}\left(k\right)\Delta_{\mathcal{V}}\left(k\right)$$

Examples of non-simple hinges include *helical hinges*, which contain both rotational and prismatic components, and *compound hinges*, such as *universal and gimbal hinges*, where the direction of the axes depends on the configuration of the hinge. We now look at specific examples of simple hinges such as the one degree of freedom rotary and prismatic hinges.

Remark 5.2 Body dynamics invariance across Newtonian frames: discussed the invariance of the dynamics of a single body in Newtonian frames, this fact extends to multibody systems as well. The dynamics of a multibody system are invariant across Newtonian frames though not across non-Newtonian ones. Observe that the velocity values enter into the dynamics equations of motion only through the a(k) and b(k) terms and any impact on the equations of motion has to happen through these quantities. Switching to a different Newtonian frame can result in a δv change in the linear velocity of the base-body. This change results in a similar additive change to the linear velocity of each body in the system, so that v(k) becomes v(k)+δv. Focusing on the inertial frame derivatives form of the equations of motion, we see that δv has no effect on the expression for aI(k) in (5.19) because it is eliminated from the v(k) − v(k+1) sub-expression. Also, the expression for the bI(k) gyroscopic force in (2.23) does not contain the linear velocity term, and hence, δv has no effect 5.1 on it either. Thus, the dynamics equations of motion are independent of the linear velocity of the base-body, and hence, they are unaffected by the specific choice of a Newtonian frame. On the other hand, both the Coriolis acceleration and gyroscopic force expressions involve the body angular velocities. Any changes in the angular velocity of the base-body – due to the use of non-Newtonian rotating frames – ripple through the angular velocities of all the bodies, changing the values of these Coriolis and gyroscopic terms and, in effect, changing the dynamics of the system.

Remark 6.1 G(k) is a generalized-inverse of H(k). It is easy to verify from the definitions in (6.13) that the following identity holds: H(k)G(k) = I (6.14) It follows therefore, that H(k)G(k)H(k) = H(k) and, hence, G(k) is a generalized inverse1 of H(k).

Remark 6.2 Nulling out of hinge accelerations. It follows from (6.15) and (6.18) that if
α+(k) = H∗(k)θ¨(k), then α(k) = 0!

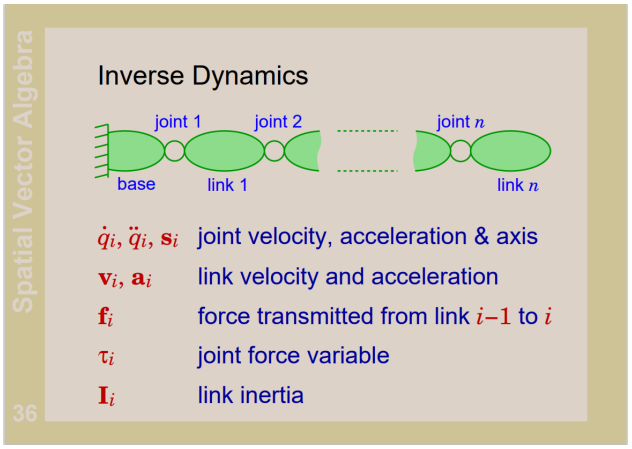
$$\mathfrak{a}^{+}(k) = H^{\star}(k)\ddot{\theta}$$

In effect, in the articulated body model, accelerations along the hinge axis *are not transmitted across* the freely articulated hinge. The only motion that is transmitted is along the other axes where there is rigid coupling. The information about this selective coupling across the hinge is contained in the τ(k) projection operator

Application to general multibody systems

In treatment used so far we assumed to work with serial chains, without loss of generality. Nonetheless equations need to be corrected for proper connection of discrete bodies, since indexing is not necessarily sequential. It is usually sufficient to adopt a part vector to map current body index with proper parent body when a tree structure exist. Is remarkable as in tree topologies a natural order do not exist since branch numbering is at least partially arbitrary at the root where ambiguity potentially exist on precedence order. Of course i possible to define some conventions, but mathematically and physically speaking there are no reasons to prefer one to another. Indeed also in serial chain a random numbering is possible, but usual convention of using sequential numbering ease understanding and debugging of algorithms.

Motion on constrained subspace interpretation



Transformations and kinematics

Key algorithms

In the following we re propose a set of algorithms taken from Jain [].
In this work they've been used for development of Newton Euler algorithm development in place of traditional algorithm more complex and extensive, and for mass matrix calculation, where maybe its effectiveness is more evident. These algorithms apply both to serial open chains and to tree-like mechanisms.

Some snatches of code illustrating basilar algorithms are reported in the following

Algorithm Recursive computation of the SKO model mass matrix

for all nodes $\mathbf{k}$ *(tips-to-base gather)*

$$\mathcal{R}(k) = \sum_{\forall i \in \mathcal{C}(k)} \mathbb{A}(k,i)\mathcal{R}(i)\mathbb{A}^*(k,i) + \mathcal{M}(k)$$

$$j = k, \quad X(k) = \mathcal{R}(k)H^*(k), \quad \mathcal{M}(k,k) = H(k)X(k)$$

while j

$$l = \wp(j)$$

$$X(l) = \phi(l,j)X(j)$$

$$\mathcal{M}(l,k) = \mathcal{M}^*(k,l) = H(l)X(l)$$

$$j = l$$

end loop

end loop

Algorithm Newton–Euler inverse dynamics algorithm for an SKO model

for all nodes k *(base-to-tips scatter)*

$$\mathcal{V}(k) = \mathbb{A}^*(\wp(k),k)\mathcal{V}(\wp(k)) + H^*(k)\dot{\theta}(k)$$
$$\alpha(k) = \mathbb{A}^*(\wp(k),k)\alpha(\wp(k)) + H^*(k)\ddot{\theta}(k) + \mathfrak{a}(k)$$

end loop

for all nodes k *(tips-to-base gather)*

$$f(k) = \sum_{\forall j \in \mathfrak{C}(k)} \mathbb{A}(k,j) f(j) + M(k)\alpha(k) + \mathfrak{b}(k)$$
$$\mathcal{T}(k) = H(k) f(k)$$

end loop

Algorithm Newton–Euler inverse dynamics algorithm

$$\mathcal{V}(n+1) = \mathbf{0}, \quad \alpha(n+1) = \mathbf{0}$$

for $k = n \cdots 1$

$$\mathcal{V}(k) = \phi^*(k+1,k)\mathcal{V}(k+1) + H^*(k)\dot{\theta}(k)$$
$$\alpha(k) = \phi^*(k+1,k)\alpha(k+1) + H^*(k)\ddot{\theta}(k) + \mathfrak{a}(k)$$

end loop

$$f(0) = \mathbf{0}$$

for $k = 1 \cdots n$

$$f(k) = \phi(k,k-1) f(k-1) + M(k)\alpha(k) + \mathfrak{b}(k)$$
$$\mathcal{T}(k) = H(k) f(k)$$

end loop

Algorithm Recursive algorithm for the articulated body inertias

$$\mathcal{P}^+(0) = \mathbf{0}$$

for $k = 1 \cdots n$

$$\mathcal{P}(k) = \phi(k,k-1)\mathcal{P}^+(k-1)\phi^*(k,k-1) + M(k)$$
$$\mathcal{D}(k) = H(k)\mathcal{P}(k)H^*(k)$$
$$\mathcal{G}(k) = \mathcal{P}(k)H^*(k)\mathcal{D}^{-1}(k)$$
$$\bar{\tau}(k) = \mathbf{I} - \mathcal{G}(k)H(k)$$
$$\mathcal{P}^+(k) = \bar{\tau}(k)\mathcal{P}(k)$$

end loop

Remark 6.4 P(k) is not a spatial inertia. The computations for the articulated body inertias are analogous to those that determine the composite spatial inertia matrix in Sect. 4.1.2. However, there is a fundamental difference. The composite spatial inertia matrix can be parametrized 6.3 Articulated Body Model Force Decomposition 107 by ten parameters. These parameters corresponded to the mass, mass center, and the rotational inertia of the composite body outboard of hinge k. However, no such reduced parametrization of the articulated body inertias is possible. The main reason for this is that P(k), while inertia-like, is not a spatial inertia. And thus, the property of a parametrization by only ten parameters, and the resultant structure, is lost. The articulated body inertias are fully populated 6 × 6 symmetric matrices requiring a full parametrization by 21 parameters.

A coordinate-frame representation of M(z) is a 6×6 symmetric, positive semidefinite matrix. While, in general, 21 elements are required to specify a 6 × 6 symmetric matrix, only ten elements are required to specify a spatial inertia matrix because of its special structure: one for the **mass scalar, three for the center of mass location 3-vector, and six for the symmetric rotational inertia matrix**.

Algorithm The mass matrix for flexible-link systems

$$
\begin{aligned}
&\mathcal{R}(0) = \mathbf{0} \\
&\textbf{for } k = 1 \cdots n \\
&\quad \mathcal{R}(k) = \Phi_{fl}(k,k-1)\mathcal{R}(k-1)\Phi_{fl}{}^{*}(k,k-1) + M_{fl}(k) \\
&\qquad = \mathcal{A}_{fl}(k)\phi(\mathbb{O}^{+}_{k-1},k-1)\mathcal{R}^{rr}(k-1)\phi^{*}(\mathbb{O}^{+}_{k-1},k-1)\mathcal{A}^{*}_{fl}(k) \\
&\qquad\quad + M_{fl}(k) \\
&\quad X(k) = \mathcal{R}(k)H^{*}_{fl}(k) \\
&\quad \mathcal{M}_{fl}(k,k) = H_{fl}(k)X(k) \\
&\qquad \left\{ \begin{aligned}
&\textbf{for } j = (k+1) \cdots n \\
&\qquad X(j) = \Phi_{fl}(j,j-1)X(j-1) \\
&\qquad\quad = \mathcal{A}_{fl}(j)\phi(\mathbb{O}^{+}_{j-1},j-1)X^{r}(j-1) \\
&\qquad \mathcal{M}_{fl}(j,k) = \mathcal{M}_{fl}{}^{*}(k,j) = H_{fl}(j)X(j) \\
&\textbf{end loop}
\end{aligned} \right. \\
&\textbf{end loop}
\end{aligned}
$$

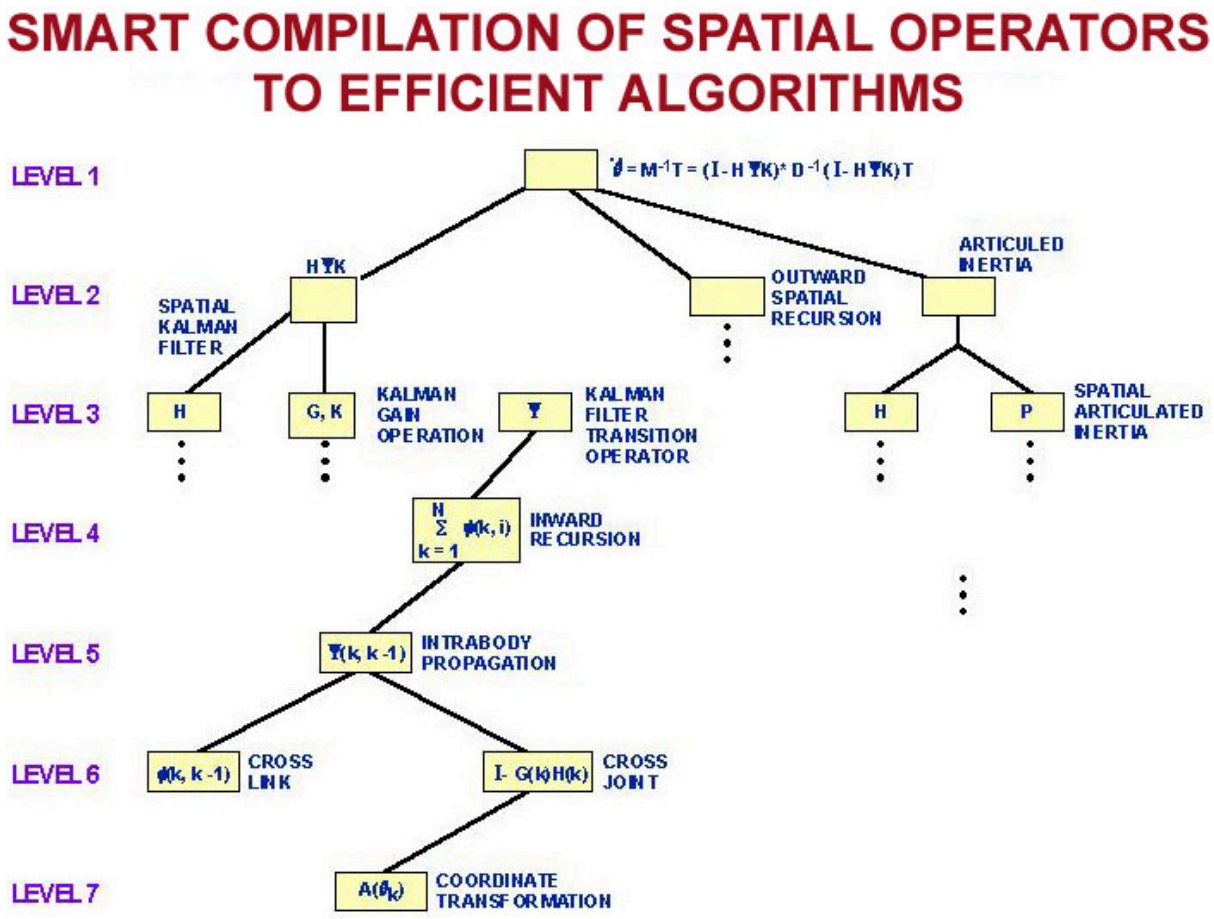
Algorithm Inverse dynamics for flexible-link systems

$$
\begin{cases}
\mathcal{V}_{fl}(n+1) = 0, \quad \alpha_{fl}(n+1) = \mathbf{0} \\
\textbf{for } k = n \cdots 1 \\
\quad \mathcal{V}^{f}_{fl}(k) = \dot{\eta}(k) \\
\quad \mathcal{V}^{r}_{fl}(k) = \phi^{*}(\mathbb{O}^{+}_{k}, k)\mathcal{A}^{*}_{fl}(k+1)\mathcal{V}_{fl}(k+1) \\
\qquad\qquad + H^{*}_{\mathbb{B}}(k)\dot{\theta}(k) - \Pi_{\mathbb{B}}(\mathbb{O}_{k})\dot{\eta}(k) \\
\quad \alpha^{f}_{fl}(k) = \ddot{\eta}(k) \\
\quad \alpha^{r}_{fl}(k) = \phi^{*}(\mathbb{O}^{+}_{k}, k)\mathcal{A}^{*}_{fl}(k+1)\alpha_{fl}(k+1) \\
\qquad\qquad + H^{*}_{\mathbb{B}}(k)\ddot{\theta}(k) - \Pi_{\mathbb{B}}(\mathbb{O}_{k})\ddot{\eta}(k) + \mathfrak{a}^{r}_{fl}(k) \\
\textbf{end loop}
\end{cases}
$$

$$
\begin{cases}
\mathfrak{f}_{fl}(0) = \mathbf{0} \\
\textbf{for } k = 1 \cdots n \\
\quad \mathfrak{f}_{fl}(k) = \mathcal{A}_{fl}(k)\phi(\mathbb{O}^{+}_{k-1}, k-1)\mathfrak{f}^{r}_{fl}(k-1) + M_{fl}(k)\alpha_{fl}(k) \\
\qquad\qquad + \mathfrak{b}_{fl}(k) + \mathfrak{K}(k)\vartheta(k) \\
\quad \mathcal{T}_{fl}(k) = \begin{bmatrix} \mathcal{T}^{f}_{fl}(k) \\ \mathcal{T}^{r}_{fl}(k) \end{bmatrix} = \begin{bmatrix} \mathfrak{f}^{f}_{fl}(k) - \Pi^{*}_{\mathbb{B}}(\mathbb{O}_{k})\mathfrak{f}^{r}_{fl}(k) \\ H_{\mathbb{B}}(k)\mathfrak{f}^{r}_{fl}(k) \end{bmatrix} \\
\textbf{end loop}
\end{cases}
$$

THE SPATIAL OPERATOR ALGEBRA LEADS TO A HIERARCHICAL SOFTWARE ARCHITECTURE BUILT FROM STANDARDIZED, REUSABLE MODULES

The Spatial Operator Algebra is easily implemented using modern software development methods. At any given level in the hierarchy, the computer automatically *decomposes* each spatial operator into a set of a few more detailed operations at the next lower level in the hierarchy. There is a corresponding decomposition in the software architecture. This process of decomposition can be viewed as that of "smart" compilation, in the sense that the compiler is made smart by built-in knowledge about the system dynamical model embedded in the spatial operators. The software modules which constitute the architecture are simple, *standardized*, and easily debugged



Illustrated the software architecture. At the highest level (Level 1 in the figure) of abstraction, the user inputs a mathematical statement. This input requires that the program decompose the operators H, J, and K that appear at this level. This decomposition occurs at Level 2 in the hierarchy. At the next level, labeled 3 in the figure, the computer recognizes that j is what is referred to as a Kalman filter transition operator. This in turn implies that j can be mechanized using an inward Kalman filtering recursion, shown at Level 4, from the tip of the

system to the base. Subsequent levels are decomposed in the same fashion. At the lowest level, Level 7 in the figure, are the very detailed coordinate transformations for reference frames attached to the various bodies constituting the multibody system.

This programming approach achieves a very high level of abstraction. The number of symbols visible to the analyst at any level in the hierarchy is very small. This means that the corresponding computer programs are also very simple. The programs are modular and map to a modular software architecture. The programs are also to a large extent system-independent, in the sense that going from one system to another is easy to do. Although the programs are simple, computational efficiency is not lost. Embedded in the programs are very efficient computational algorithms.