Dynamics and Real-Time Simulation (DARTS) Laboratory

Spatial Operator Algebra (SOA)

8. Graph Theory Structure

Abhinandan Jain

June 19, 2024

https://dartslab.jpl.nasa.gov/

Jet Propulsion Laboratory
California Institute of Technology

SOA Foundations Track Topics (serial-chain rigid body systems)

1. **Spatial (6D) notation** – spatial velocities, forces, inertias; spatial cross-product, rigid body transformations & properties; parallel axis theorem
2. **Single rigid body dynamics** – equations of motion about arbitrary frame using spatial notation
3. **Serial-chain kinematics** – minimal coordinate formulation, hinges, velocity recursions, Jacobians; first spatial operators; O(N) scatter and gather recursions
4. **Serial-chain dynamics** – equations of motion using spatial operators; Newton–Euler mass matrix factorization; O(N) inverse dynamics; composite rigid body inertia; forward Lyapunov equation; mass matrix decomposition; mass matrix computation; alternative inverse dynamics
5. **Articulated body inertia -** Concept and definition; Riccati equation; alternative force decompositions
6. **Mass matrix factorization and inversion** – spatial operator identities; Innovations factorization of the mass matrix; Inversion of the mass matrix
7. **Recursive forward dynamics** – O(N) recursive forward dynamics algorithm; including gravity and external forces; inter-body forces identity

See https://dartslab.jpl.nasa.gov/References/index.php for publications and references on the SOA methodology.

SOA Generalization Track Topics

8. **Graph theory based structure** – BWA matrices, connection to multibody systems
9. **Tree topology systems** – generalization to tree topology rigid body systems, SKO/SPO operators, gather/scatter algorithms
10. **Closed-chain dynamics (cut-joint)** – holonomic and non-holonomic constraints, cut-joint method, operational space inertia, projected dynamics
11. **Closed-chain dynamics (constraint embedding)** – constraint embedding for graph transformation, minimal coordinate closed-chain dynamics
12. **Flexible body dynamics** – Extension to flexible bodies, modal representations, recursive flexible body dynamics

Recap

Previous Session Recap

- Used the operator expression for the mass matrix inverse to develop the O(N) ATBI forward dynamics algorithm
- Described simple way to obtain inter-body forces if desired
- Developed extensions for handling gravity and external forces
- Developed O(N) generalized hybrid dynamics algorithm
  - Elegant combination of ATBI forward dynamics and CRB inverse dynamics

Serial chain summary

$\mathcal{E}_\phi$ is nilpotent

- Every power of $\mathcal{E}_\phi$ results in a matrix with the sub-diagonal shifted one step lower

$$\mathcal{E}_\phi \triangleq \begin{pmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \phi(2,1) & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \phi(3,2) & \dots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \phi(n,n-1) & \mathbf{0} \end{pmatrix}$$

- At the nth power, the result is zero: $\mathcal{E}_\phi^n = \mathbf{0}$
- Hence $\mathcal{E}_\phi$ is **nilpotent**!

$$\mathcal{E}_{\mathbb{A}} = \begin{pmatrix} \cdot & \cdot & \cdot & \cdot \\ X & \cdot & \cdot & \cdot \\ \cdot & X & \cdot & \cdot \\ \cdot & \cdot & X & \cdot \end{pmatrix}, \quad \mathcal{E}_{\mathbb{A}}^2 = \begin{pmatrix} \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ X & \cdot & \cdot & \cdot \\ \cdot & X & \cdot & \cdot \end{pmatrix}, \quad \mathcal{E}_{\mathbb{A}}^3 = \begin{pmatrix} \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ X & \cdot & \cdot & \cdot \end{pmatrix}$$

System level equations of motion

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

$$\boxed{\begin{aligned} \mathcal{V} &= \phi^* H^* \dot{\theta} \\ \alpha &= \phi^* (H^* \ddot{\theta} + \mathfrak{a}) \\ \mathfrak{f} &= \phi(\mathbf{M}\alpha + \mathfrak{b}) \\ \mathcal{T} &= H\mathfrak{f} \end{aligned}} \Rightarrow \boxed{\begin{aligned} \mathcal{T} &= H\,\phi\,\left[\mathbf{M}\,\phi^*\,\left(H^*\,\ddot{\theta} + \mathfrak{a}\right) + \mathfrak{b}\right] \\ &= \mathcal{M}(\theta)\ddot{\theta} + \mathcal{C}(\theta, \dot{\theta}) \end{aligned}}$$

familiar **mass matrix** $\mathcal{M}(\theta) = H\phi\mathbf{M}\phi^* H^* \in \mathcal{R}^{\mathcal{N}\times\mathcal{N}}$

Newton-Euler factorization

Coriolis terms $\mathcal{C}(\theta, \dot{\theta}) \triangleq H\phi(\mathbf{M}\phi^*\mathfrak{a} + \mathfrak{b}) \in \mathcal{R}^{\mathcal{N}}$

Later – These equations of motion hold for any tree/branched system.

Mass Matrix Factorization & Inversion

$$\mathcal{M} = \mathrm{H}\phi\mathrm{M}\phi^*\mathrm{H}^*$$

Analytical Newton-Euler factorization of the mass matrix

$$\mathcal{M} = [\mathrm{I} + \mathrm{H}\phi\mathcal{K}]\mathcal{D}[\mathrm{I} + \mathrm{H}\phi\mathcal{K}]^*$$

Analytical Innovations factorization of the mass matrix

$$[\mathrm{I} + \mathrm{H}\phi\mathcal{K}]^{-1} = \mathrm{I} - \mathrm{H}\psi\mathcal{K}$$

$$\boxed{\mathcal{M}^{-1} = [\mathbf{I} - \mathrm{H}\psi\mathcal{K}]^* \mathcal{D}^{-1} [\mathbf{I} - \mathrm{H}\psi\mathcal{K}]}$$

Analytical operator expression for the mass matrix inverse

Spatial operator analysis

- Velocity expression
- Jacobian
- Mass matrix NE factorization
- Lyapunov equation for CRBs
- Mass matrix decomposition
- Riccati equation for ATBI
- Several operator identities
- Mass matrix Innovations factorization
- Mass matrix determinant
- Mass matrix inverse and factorization

Have started to build up a vocabulary of spatial operators that can be used to express and manipulate the structure of dynamics quantities.

Illustrates the rationale for the ***algebra*** *part of SOA from the analytical transformations and simplifications possible using the operators.*

$$\begin{tikzcd}
 & \mathcal{E}_\phi & \mathcal{H} \arrow[d] & \mathcal{M} & \\
 & & \phi \arrow[drr] \arrow[dd] & & \\
 & & & & \mathcal{R} \\
 & & \mathcal{P} \arrow[d] & & \\
\mathcal{D} & \mathcal{G} & \mathcal{K} & \tau & \mathcal{E}_\psi \arrow[d] \\
 & & & & \psi
\end{tikzcd}$$

spatial operators family

Recursive Computational Algorithms

- O(N) Gather and scatter recursions pattern
- O(N) Body velocities scatter recursion
- O(N) CRBs gather recursion
- $O(\mathcal{N}^2)$ mass matrix computation
- O(N) NE scatter/gather inverse dynamics
- $O(\mathcal{N}^2)$ inverse dynamics based mass matrix
- O(N) CRBs based inverse dynamics
- O(N) ATBI gather recursion
- $O(\mathcal{N}^2)$ forward dynamics
- O(N) ATBI forward dynamics
- O(N) hybrid dynamics

Can derive such low-cost scatter/gather algorithms usually by examination of the spatial operator expressions.

Generalization to other multibody systems

Generalization to non-serial, non-rigid multibody systems

- So far we have focused on serial-chain, rigid body systems
- We would like to generalize to the more complex
  - Non-serial topology systems
  - Non-rigid body systems
- Since each such system is sufficiently different in the basic dynamics, the process has been to
  - Set up the equations of motion for the specific system
  - Define the appropriate spatial operators
  - Carry out the operator based analysis to develop the corresponding expressions and algorithms
- This has been done successfully for the broad class of multibody systems leading to mass matrix inverse expressions as well as recursive ATBI like algorithms

Diverse, but similar SOA solutions

- In reality, multibody systems are often a mix of different types
    - Rigid/flex bodies
    - Regular and flex joints
    - Geared joints
    - Branching and loops
    - Prescribed and non-prescribed motion
- It is daunting – and tedious – to repeat the formulation, analysis, and algorithm development process over and over for the many different combinations
- Given the common patterns that were clear in the SOA based mass matrix factorization inversion and algorithm development across the different types we have the question:

What are the common patterns and properties that make the SOA process work across such a broad family of system types?

Common multibody formulation patterns

What are the common patterns and properties that make the SOA process work across such a broad family of system types?

- If we can identify such common patterns, then it would considerably simplify the analysis and algorithms development burden
- For any new multibody system, we would be able to simply check if its dynamics formulation met the requirements, and if so, we would automatically be guaranteed that the analysis and algorithms were applicable.
- Furthermore, this would also define the essential properties that need to be satisfied, and so provide a goal for the upfront formulation process to try and meet (eg. constraint embedding).

Need to develop an abstraction layer

- Towards this goal, we will take a step back from the multibody formulation specifics
- We will develop new ideas and concepts to help develop an abstraction layer to capture the common underlying structure
- For this, we turn to ***graph theory*** for mathematical tools and language
- The graph theory tools will
  - help us avoid starting from square one every time
  - help define what are the minimal requirements for the theory and algorithms to apply
  - help define what do we need to generalize
- We will find that much of the analysis and algorithm investment we made for the “simple” serial-chain, rigid body case carries forward much more broadly

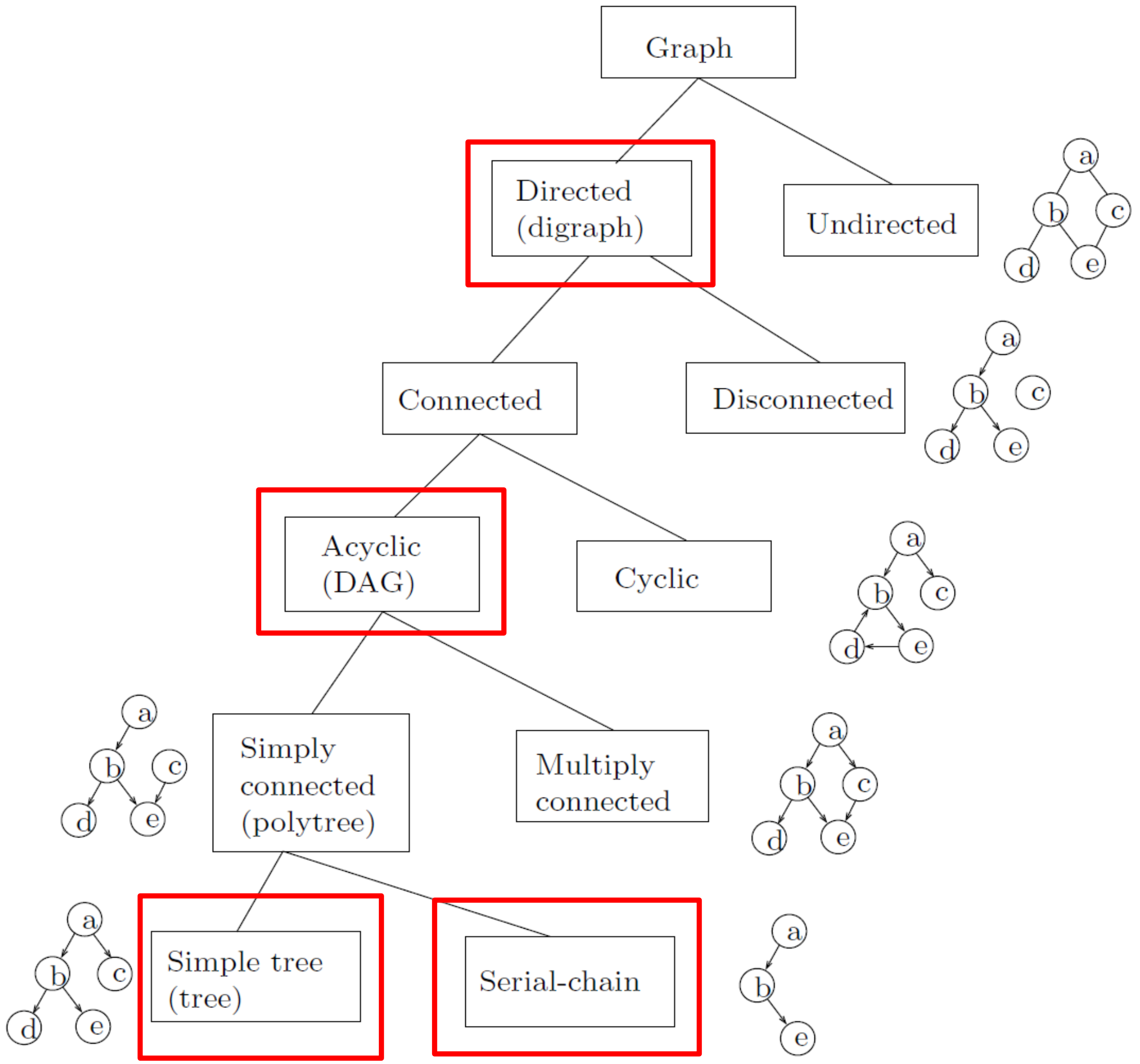
Multibody dynamics and graph theory

- Graph methods have been used in multibody context primarily for bookkeeping the topological structure, and sometimes for kinematic analysis
- We will take this further and use analytical methods from graph theory and apply to multibody problems.

Graph Theory Background

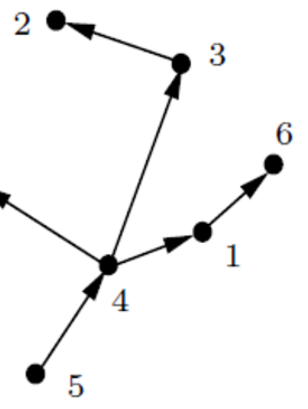
Graphs taxonomy

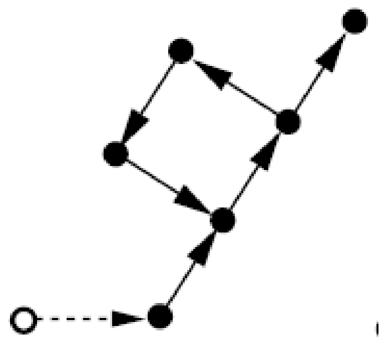
- Graphs are made up of **nodes** and **edges**
- Directed graph (**digraph**) edges have direction
- Directed, acyclic graphs (DAGs)
- **Trees**: nodes have unique parent
- **Serial-chain**; nodes have unique children



Some definitions for digraphs

- $\wp(k)$- set of **parent** nodes for the kth node
- $\mathcal{C}(k)$- set of **children** nodes for the kth node
- **root** node – a node with no parents
- **tip** node – a node with no children
- $i \prec j$ - if node j is an ancestor of node i
- $i \nprec j$ - if node j is not an ancestor of node i
- Nodes i & j are a **related** node pair if there is a path connecting the pair of nodes
- A **tree** has a single root node and one or more tip nodes

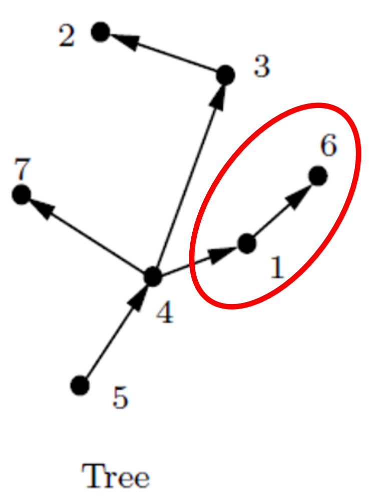


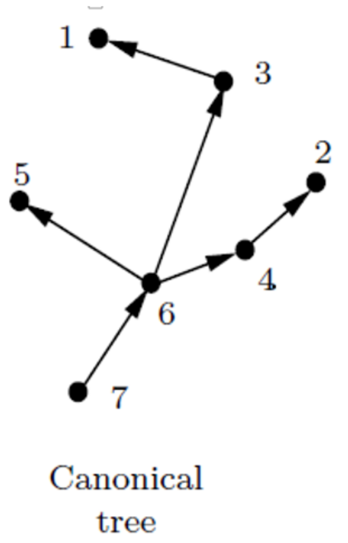


Canonical trees

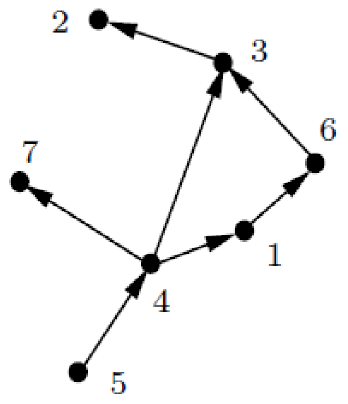
For a tree, a node has at most a single parent node

- There is a partial ordering among the nodes
- When we use an indexing scheme consistent with the partial order, i.e parent has index greater than that of child – the tree is said to be **canonical**
- We will not rely much on canonical indexing
- The canonical concept does not apply to general graphs since there is no partial ordering available



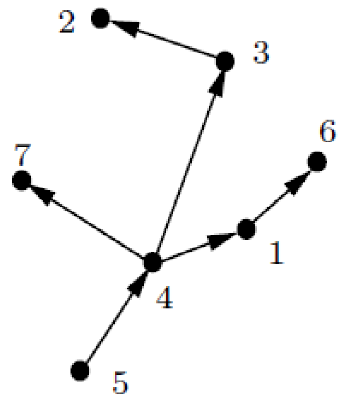


Directed acyclic graph (DAG) examples



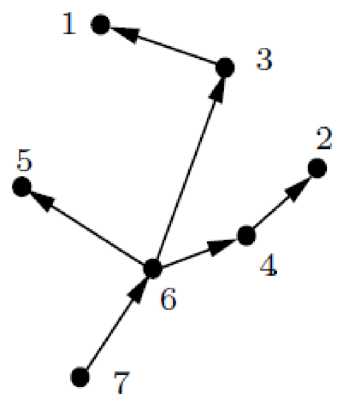
DAG

multiple paths connecting a pair of nodes



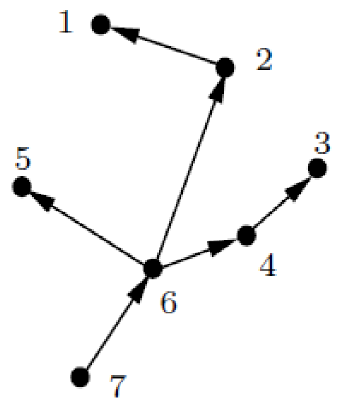
Tree

only single path at most connecting a pair of nodes



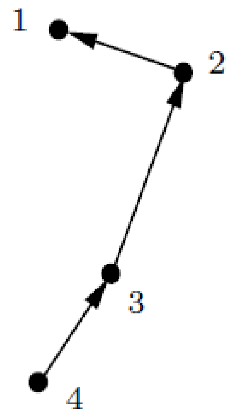
Canonical tree

child index is smaller than ancestor index



Strictly canonical tree

index within branch are canonical



Canonical serial chain

our focus so far

- Tree nodes are only partially ordered
- Serial-chain nodes have strict ordering
- Can get canonical property by renumbering

Graph adjacency matrix

Digraph adjacency matrix

The **adjacency matrix** for a digraph with n nodes is an nxn (nodes vs nodes) square matrix

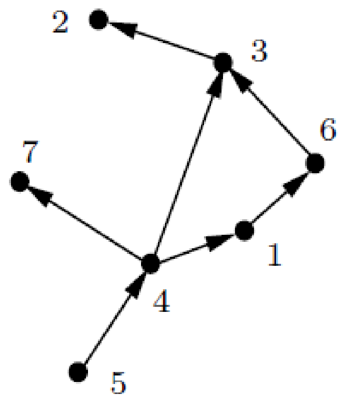
- The rows and columns are each indexed by the set of nodes
- A matrix element is 1 when the row index node is a direct parent of the column index node, and 0 otherwise
- The adjacency matrix fully describes the connectivity topology for a graph

An **incidence matrix** (nodes vs edges) is an alternative representation of a graph’s topology.

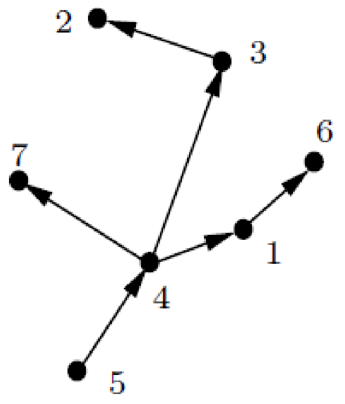
Properties of an adjacency matrix

- Non-zero entries along a row represent children nodes
- Non-zero entries along a column represent parent nodes
  - For a tree, there can be only a single non-zero entry in a column (unique parent)
  - A row can have multiple entries for children
- Since a node cannot be its immediate child and parent, the diagonal has 0s
- Similarly, in a tree, a pair of nodes cannot be each others parent and child simultaneously, and hence an adjacency matrix and its transpose are disjoint (i.e. no common non-zero members)

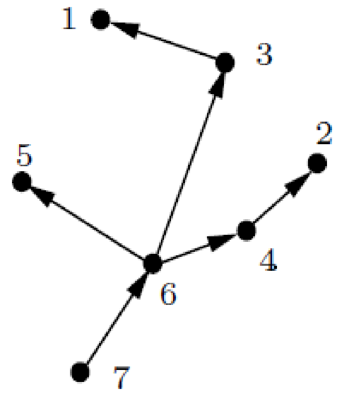
Adjacency matrix examples



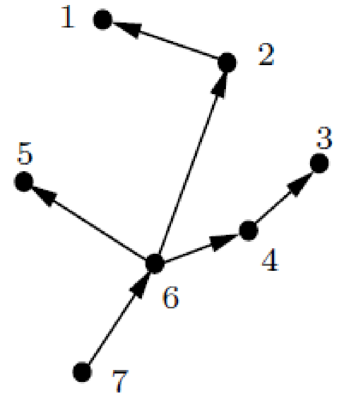
DAG



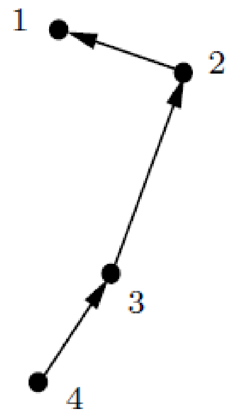
Tree



Canonical tree



Strictly canonical tree



Canonical serial chain

Adjacency matrices for these trees

$$\begin{pmatrix} . & . & . & . & . & 1 & . \\ . & . & . & . & . & . & . \\ . & 1 & . & . & . & . & . \\ 1 & . & 1 & . & . & . & 1 \\ . & . & . & 1 & . & . & . \\ . & . & 1 & . & . & . & . \\ . & . & . & . & . & . & . \end{pmatrix}$$

DAG

$$\begin{pmatrix} . & . & . & . & . & 1 & . \\ . & . & . & . & . & . & . \\ . & 1 & . & . & . & . & . \\ 1 & . & 1 & . & . & . & 1 \\ . & . & . & 1 & . & . & . \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \end{pmatrix}$$

Tree

$$\begin{pmatrix} . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ 1 & . & . & . & . & . & . \\ . & 1 & . & . & . & . & . \\ . & . & . & . & . & . & . \\ . & . & 1 & 1 & 1 & . & . \\ . & . & . & . & . & 1 & . \end{pmatrix}$$

Canonical tree

$$\begin{pmatrix} . & . & . & . & . & . & . \\ 1 & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ . & . & 1 & . & . & . & . \\ . & . & . & . & . & . & . \\ . & 1 & . & 1 & 1 & . & . \\ . & . & . & . & . & 1 & . \end{pmatrix}$$

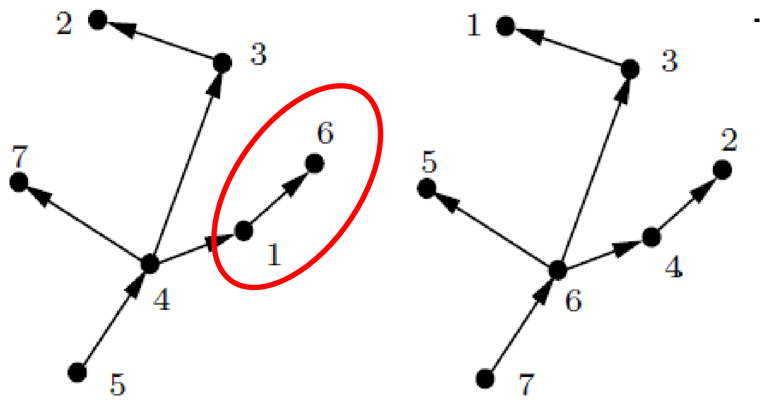
Strictly canonical tree

$$\begin{pmatrix} . & . & . & . \\ 1 & . & . & . \\ . & 1 & . & . \\ . & . & 1 & . \end{pmatrix}$$

Canonical serial-chain

Properties of an adjacency matrix (contd)

- The adjacency matrix is strictly lower-triangular for a canonical tree
- While helpful for intuition, we will not be needing or requiring the canonical property



Tree

Canonical tree

$$\begin{pmatrix} \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot \\ 1 & \cdot & 1 & \cdot & \cdot & \cdot & 1 \\ \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{pmatrix} \begin{pmatrix} \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 1 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & 1 & 1 & 1 & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot \end{pmatrix}$$

Adjacency matrix expressions

- The elements of an adjacency matrix can be expressed as

$$\boxed{\mathbb{S}(\mathrm{r},\mathrm{q}) = \mathbb{1}_{[\mathrm{r}\in\wp(\mathrm{q})]} = \mathbb{1}_{[\mathrm{q}\in\mathsf{C}(\mathrm{r})]}}$$

where $\boxed{\mathbb{1}_{[<\mathrm{cond}>]}}$ denotes the **indicator function** whose value is 1 of the condition is true, and 0 otherwise

- For a tree, this simplifies to (since unique parent)

$$\boxed{\mathbb{S}(\mathrm{r},\mathrm{q}) = \delta_{\mathrm{r},\wp(\mathrm{q})}}$$

Weighted adjacency matrix

Weighted adjacency matrix

- A weighted adjacency matrix is one where the 1 entries are replaced with w(i, j) non-zero weight values
- Instead of

$$\boxed{\mathbb{S}(r, q) = \mathbb{1}_{[r\in\wp(q)]} = \mathbb{1}_{[q\in\mathbf{C}(r)]}}$$

  we have

$$\boxed{\mathbb{S}_W(k, j) = \underset{\text{weight}}{w}(k, j)\ \mathbb{1}_{[k\in\wp(j)]}}$$

- For a tree graph, this simplifies to

$$\boxed{\mathbb{S}_W(r, q) = \delta_{r,\wp(q)} w(r, q)}$$

Tensor notation

- Tensor notation simplifies notation by avoiding the need for explicitly showing summations
- For a matrix product $\boxed{Y = AB}$

$$Y(i,j) = \sum_{r=1}^{n} A(i,r)\, B(r,j) \;= \boxed{A(i,r)\, B(r,j)}$$

tensor notation

- For a triple product $\boxed{Y = ABC}$

$$Y(i,j) = A(i,r)\, B(r,s)\, C(s,j)$$

with <u>implicit double summation</u> over the repeated r and s indices

Square of a tree adjacency matrix

Claim:

non-zero only when r is ***grandparent*** *of q*

$$\mathbb{S}^2_W(\mathrm{r},\mathrm{q}) = \delta_{\mathrm{r},\wp^2(\mathrm{q})}\left[w\left(\wp^2(\mathrm{q}),\wp(\mathrm{q})\right) * w\left(\wp(\mathrm{q}),\mathrm{q}\right)\right]$$

product of weights for the connecting edge pair

Derivation: *(using tensor notation)*

$$\begin{aligned}\mathbb{S}^2_W(\mathrm{r},\mathrm{q}) &= \mathbb{S}_W(\mathrm{r},\mathrm{s})\,\mathbb{S}_W(\mathrm{s},\mathrm{q})\\ &\overset{8.13}{=} \delta_{\mathrm{r},\wp(\mathrm{s})}w(\mathrm{r},\mathrm{s}) * \delta_{\mathrm{s},\wp(\mathrm{q})}w(\mathrm{s},\mathrm{q})\\ &= \delta_{\mathrm{r},\wp^2(\mathrm{q})}\left[w\left(\wp^2(\mathrm{q}),\wp(\mathrm{q})\right) * w\left(\wp(\mathrm{q}),\mathrm{q}\right)\right]\end{aligned}$$

General powers of a tree adjacency matrix

Claim:

non-zero only when r is k-level ancestor of q

$$\mathbb{S}_W^k(\mathrm{r},\mathrm{q}) = \delta_{\mathrm{r},\wp^k(\mathrm{q})}\left[w\left(\wp^k(\mathrm{q}),\wp^{k-1}(\mathrm{q})\right) * \cdots * w\left(\wp(\mathrm{q}),\mathrm{q}\right)\right]$$

product of weights along the connecting path

SHOW!
(use induction)

Comments:

The only non-zero elements of the kth power $\mathbb{S}_W^k(\mathrm{r},\mathrm{q})$ are for nodes r and q that are exactly <u>k nodes distance apart</u> in the graph!

Nilpotency of a tree weighted adjacency matrix

- The only non-zero elements of the kth power $\mathbb{S}_W^k(r, q)$ are for the r and q nodes that are exactly k nodes apart in the graph!
- For a tree system with n nodes, the nodes can be at most (n-1) nodes part, and hence the nth power of the weighted adjacency matrix has all zero elements!
  - Thus the weighted adjacency matrix for a tree is **nilpotent**!

What about non-tree graph systems?

- The kth power of the weighted adjacency matrix contains entries with path length k.
- If the graph has **loops** and is not a tree, a node in a loop can be be its own ancestor when we go around a loop, and this means
  - that higher powers can have non-zero elements along the diagonal
  - also that the higher powers may never vanish
- Thus the weighted adjacency matrix may not be nilpotent for non-tree systems

Next steps …

Recall: $\mathcal{E}_\phi$ nilpotent operator

- Every power of $\mathcal{E}_\phi$ results in a matrix with the sub-diagonal shifted one step lower

$$\mathcal{E}_\phi \triangleq \begin{pmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \phi(2,1) & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \phi(3,2) & \dots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \phi(n,n-1) & \mathbf{0} \end{pmatrix}$$

- At the nth power, the result is zero: $\mathcal{E}_\phi^n = \mathbf{0}$
- Hence $\mathcal{E}_\phi$ is **nilpotent**!

$$\mathcal{E}_{\mathbb{A}} = \begin{pmatrix} \cdot & \cdot & \cdot & \cdot \\ X & \cdot & \cdot & \cdot \\ \cdot & X & \cdot & \cdot \\ \cdot & \cdot & X & \cdot \end{pmatrix}, \quad \mathcal{E}_{\mathbb{A}}^2 = \begin{pmatrix} \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ X & \cdot & \cdot & \cdot \\ \cdot & X & \cdot & \cdot \end{pmatrix}, \quad \mathcal{E}_{\mathbb{A}}^3 = \begin{pmatrix} \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ X & \cdot & \cdot & \cdot \end{pmatrix}$$

Where are we going with this?

- Previously our multibody development was specific to serial chains, and our nilpotency based development was based on the structure specific to serial-chains
  - Also the $\mathcal{E}_\phi$ nilpotent matrix had square, block matrix elements and lower-triangular structure
- Currently we are looking at adjacency matrices for general tree systems
  - Have scalar elements, but have generalized to have weight elements instead of just 1 and 0 elements
- Our future steps are to
  - generalize the graph adjacency matrices to ones with block matrix elements
  - then circle back and connect up the adjacency matrix development with multibody system analysis and SOA operators

Block-Weighted Adjacency (BWA) Matrix

Generalizing to matrix elements

- So far the adjacency matrices have scalar elements
- We now generalize the adjacency matrices to have block matrix elements – and refer to these as **Block-Weighted Adjacency (BWA)** matrices
- Assign a row dimension (denoted m_k for the kth row) to all rows. This is also the dimension of the kth column.

General BWA matrix elements

- The elements of the BWA matrix can be expressed as follows:

$$\mathbb{S}_{\mathcal{W}}(k,j) = \boxed{w(k,j)}\, \mathbb{1}_{[k\in\wp(j)]} = \begin{cases} w(k,j) & \text{if} \quad k \in \wp(j) \\ \mathbf{0} & \text{otherwise} \end{cases} \in \mathcal{R}^{m_k \times m_j}$$

- The matrix elements can be non-square
- The non-zero block entries are in the same locations as the graph's adjacency matrix
- The BWA matrix is square with dimension

$$N \triangleq \sum_{k=1}^{n} m_k$$

Tree BWA matrix elements

For a tree system, each node can have at most one parent node, and so the matrix elements expression

$$\mathbb{S}_W(k, j) = w(k, j)\ \mathbb{1}_{[k \in \wp(j)]}$$

simplifies to:

$$\mathbb{S}_W(r, q) = \delta_{r, \wp(q)} w(r, q)$$

Powers of a tree BWA

Previously, for weighted adjacency matrices

$$\mathbb{S}^k_W(r,q) = \delta_{r,\wp^k(q)}\left[w\left(\wp^k(q),\wp^{k-1}(q)\right) * \cdots * w\left(\wp(q),q\right)\right]$$

non-zero (scalar) only when r is k-level ancestor of q

Claim:

$$\mathbb{S}^k_W(r,q) = \delta_{r,\wp^k(q)}\left[w\left(\wp^k(q),\wp^{k-1}(q)\right) * \cdots * w\left(\wp(q),q\right)\right] \in \boxed{\mathcal{R}^{m_r \times m_q}}$$

matrix element

SHOW!

Comments:

- The structural results from the scalar elements generalize entirely to matrix elements
- The only <u>non-zero elements</u> of the kth power $\mathbb{S}^k_W(r,q)$ are nodes r and q that are k nodes apart in the graph!

Nilpotency of a tree BWA matrix

- Once again, the only non-zero elements of the kth power $\mathbb{S}_{W}^{k}(r, q)$ are nodes r and q that are k nodes apart in the graph!
- For a tree system with n nodes, the nodes can be at most (n-1) nodes part, and hence the nth power has all zero elements!
    - Thus the BWA matrix for a tree is **nilpotent**!

1-resolvent of a Tree BWA

Recall: Nilpotent matrices & 1-resolvents

- A square matrix $\mathcal{U}$ is said to be nilpotent if one of its powers becomes 0, i.e. if for some n

$$\mathcal{U}^n = \mathbf{0}$$

- For a nilpotent $\mathcal{U}$, we have

$$\underbrace{(\mathbf{I} - \mathcal{U})^{-1}}_{\text{1-resolvent}} = \mathbf{I} + \mathcal{U} + \mathcal{U}^2 + \cdots + \mathcal{U}^{n-1}$$

*Series expansion truncates after only a **finite** number of terms for nilpotent matrix, hence the 1-resolvent inverse is well defined*

1-resolvent of a Tree BWA matrix

- Tree BWA matrices are nilpotent
- Hence the 1-resolvent is well defined and given by:

$$(\mathbf{I} - \mathbb{S}_W)^{-1} = \mathbf{I} + \mathbb{S}_W + \mathbb{S}_W^2 + \cdots + \mathbb{S}_W^{n-1}$$

all disjoint

- This is a general result - no requirement that the BWA matrix be lower-triangular or only have a major sub-diagonal (as earlier for serial-chains case).
  - This applies to <u>all</u> tree BWA matrices

Recall: Familiar example of a similar 1-resolvent from multibody context

$\mathcal{E}_\phi$ is **nilpotent** for a serial chain system, and we can thus define its 1-resolvent

$$\mathcal{E}_\phi \triangleq \begin{pmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \phi(2,1) & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \phi(3,2) & \dots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \phi(n,n-1) & \mathbf{0} \end{pmatrix}$$

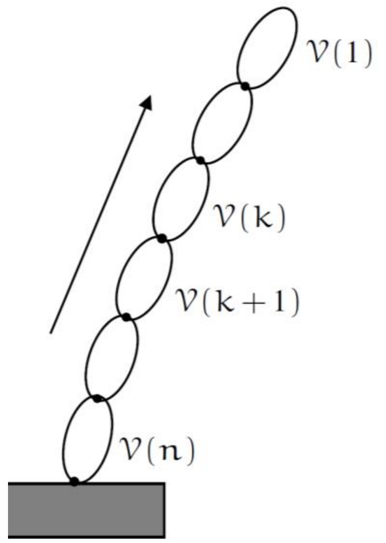
weights

$$\boxed{\phi} \triangleq (\mathbf{I}-\mathcal{E}_\phi)^{-1} = \mathbf{I} + \mathcal{E}_\phi + \mathcal{E}_\phi^2 + \cdots + \mathcal{E}_\phi^{n-1}$$

$$= \begin{pmatrix} \mathbf{I} & \mathbf{0} & \dots & \mathbf{0} \\ \phi(2,1) & \mathbf{I} & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \phi(n,1) & \phi(n,2) & \dots & \mathbf{I} \end{pmatrix}$$

Lower triangular with inter-body rigid transformation matrix elements

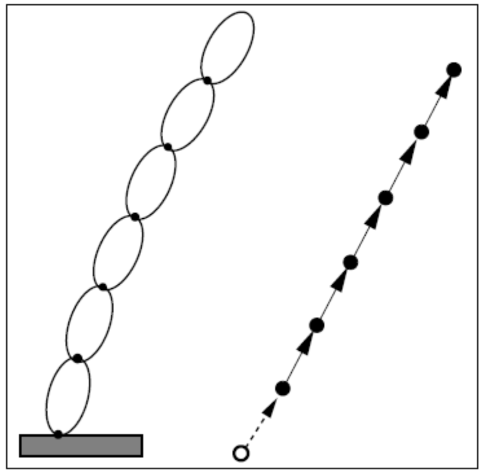
Rows: *parent body*
Columns: *child body*



Serial-chain $\mathcal{E}_\phi$ is a tree BWA matrix

Have

$$\mathcal{E}_\phi \triangleq \begin{pmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \phi(2,1) & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \phi(3,2) & \dots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \phi(n,n-1) & \mathbf{0} \end{pmatrix}$$



that is,

$$\boxed{\mathcal{E}_\phi(j,k) = \delta_{j,\wp(k)}\phi(\wp(k),k)}$$

Thus $\mathcal{E}_\phi$ is a serial-chain BWA matrix with elements

$$w(\wp(k),k) = w(k+1,k) \triangleq \phi(k+1,k) = \phi(\wp(k),k)$$

ϕ is the 1-resolvent of $\mathcal{E}_\phi$: the tree BWA matrix

From the fact that $\mathcal{E}_\phi$ is a tree BWA matrix

$$\phi \overset{\triangle}{=} (\mathbf{I} - \mathcal{E}_\phi)^{-1} = \mathbf{I} + \mathcal{E}_\phi + \mathcal{E}_\phi^2 + \cdots + \mathcal{E}_\phi^{n-1}$$

$$= \begin{pmatrix} \mathbf{I} & \mathbf{0} & \dots & \mathbf{0} \\ \phi(2,1) & \mathbf{I} & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \phi(n,1) & \phi(n,2) & \dots & \mathbf{I} \end{pmatrix}$$

Jet Propulsion Laboratory
California Institute of Technology

Notational adjustment

- We have shown that a multibody serial-chain $\mathcal{E}_\phi$ is a tree BWA matrix.
- Change notation for BWA matrices to more closely parallel notation used for serial-chain multibody systems

BWA matrix $\boxed{\mathcal{E}_{\mathbb{A}}} \equiv \mathbb{S}_W$

BWA 1-resolvent $\boxed{\mathbb{A}} \equiv (\mathbf{I} - \mathbb{S}_W)^{-1}$

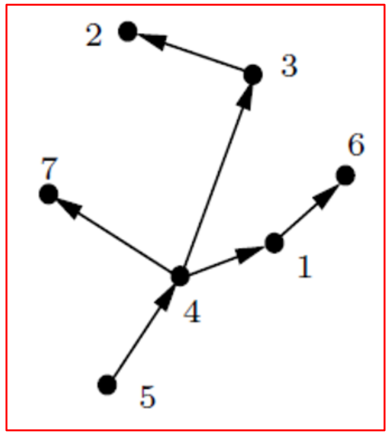
$$= \mathbf{I} + \mathbb{S}_W + \mathbb{S}_W^2 + \cdots + \mathbb{S}_W^{n-1}$$

Sparsity structure of 1-resolvent A

Claim:

$\mathbb{A}(i,j)$ is 0 if nodes i and j are <u>unrelated</u>

(i.e. there is no path connecting i & j)



Derivation:

$$\mathbb{A} = \mathrm{I} + \mathbb{S}_W + \mathbb{S}_W^2 + \cdots + \mathbb{S}_W^{n-1}$$

non-zero entries only for related nodes

The 1-resolvent A and its transpose

Have $\mathbb{A} = \mathbf{I} + \mathbb{S}_W + \mathbb{S}_W^2 + \cdots + \mathbb{S}_W^{n-1}$

Claim:

$\mathbb{A}(i, j)$ and A(j, i) cannot both be non-zero away from the diagonal

Derivation:

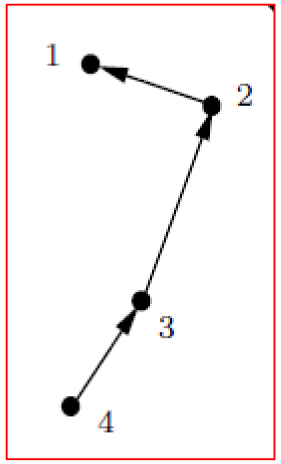
$\mathbb{A}(i, j)$ is non-zero only if i is an ancestor of j. If so, j cannot be an ancestor as well within the tree

This establishes the disjointedness property of the 1-resolvent and its transpose without appealing to a "lower triangularity" like property!

Canonical Serial-Chain example

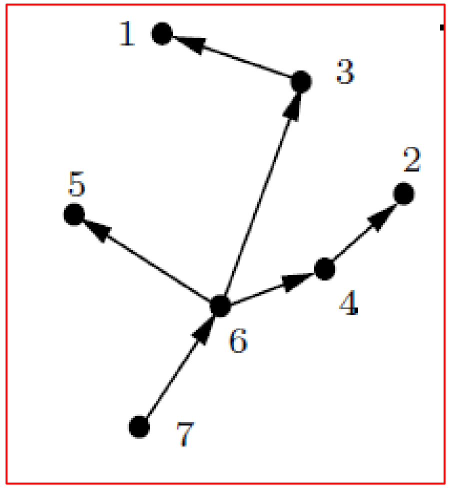
$$\mathcal{E}_{\mathbb{A}} = \begin{pmatrix} \cdot & \cdot & \cdot & \cdot \\ X & \cdot & \cdot & \cdot \\ \cdot & X & \cdot & \cdot \\ \cdot & \cdot & X & \cdot \end{pmatrix}, \quad \mathcal{E}_{\mathbb{A}}^2 = \begin{pmatrix} \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ X & \cdot & \cdot & \cdot \\ \cdot & X & \cdot & \cdot \end{pmatrix}, \quad \mathcal{E}_{\mathbb{A}}^3 = \begin{pmatrix} \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ X & \cdot & \cdot & \cdot \end{pmatrix}$$

$$\mathbb{A} = \begin{pmatrix} \mathbf{I} & \cdot & \cdot & \cdot \\ X & \mathbf{I} & \cdot & \cdot \\ X & X & \mathbf{I} & \cdot \\ X & X & X & \mathbf{I} \end{pmatrix}$$



For a serial-chain, **every pair of nodes is related**, and hence the lower-triangular part of the 1-resolvent is **fully dense**

Tree example



For a tree, not all pairs of nodes are related (eg. 3 and 5 are un-related), and there are 0 elements for all such unrelated pairs leading to **structural sparsity**.

This tree happens to be canonical, but this property is irrelevant.

$$\mathcal{E}_{\mathbb{A}} = \begin{pmatrix} \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ X & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & X & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & X & X & X & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & X & \cdot \end{pmatrix} \qquad \mathbb{A} = \begin{pmatrix} \mathbf{I} & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \mathbf{I} & \cdot & \cdot & \cdot & \cdot & \cdot \\ X & \cdot & \mathbf{I} & \cdot & \cdot & \cdot & \cdot \\ \cdot & X & \cdot & \mathbf{I} & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \mathbf{I} & \cdot & \cdot \\ X & X & X & X & X & \mathbf{I} & \cdot \\ X & X & X & X & X & X & \mathbf{I} \end{pmatrix}$$

Properties of the Tree BWA 1-resolvent

Semi-group (or chain-rule) property of elements

Claim:

node k can be arbitrary node on path connecting i & j

$$\boxed{\mathbb{A}(i,j) = \mathbb{A}(i,k)\mathbb{A}(k,j) \quad \forall\, k:\; i \succeq k \succeq j}$$

chain

Derivation:

Let $i = \wp^m(j)$ for some m, and $k = \wp^r(j)$ for some $r < m$.

$$\begin{aligned}
\mathbb{A}(i,j) &\overset{8.18}{=} w\left(\wp^m(j), \wp^{m-1}(j)\right) * \cdots * w\left(\wp(j), j\right) \\
&= \left[w\left(\wp^m(j), \wp^{m-1}(j)\right) * \cdots * w\left(\wp(k), k\right)\right] * \left[w\left(k, l\right) * \cdots * w\left(\wp(j), j\right)\right] \\
&\overset{8.18}{=} \mathbb{A}(\wp^m(j), k) * \mathbb{A}(k,j) = \mathbb{A}(i,k) * \mathbb{A}(k,j)
\end{aligned}$$

The $\tilde{\mathbb{A}}$ matrix

Define $\boxed{\tilde{\mathbb{A}} \triangleq \mathbb{A} - \mathbf{I}}$

Claim:

$$\boxed{\tilde{\mathbb{A}} = \mathbb{A} - \mathbf{I} = \mathcal{E}_{\mathbb{A}}\mathbb{A} = \mathbb{A}\mathcal{E}_{\mathbb{A}}}$$

Derivation:

Use $\mathrm{X}(\mathbf{I} - \mathrm{X})^{-1} = (\mathbf{I} - \mathrm{X})^{-1}\mathrm{X} = (\mathbf{I} - \mathrm{X})^{-1} - \mathbf{I}$

with $\mathrm{X} = \mathcal{E}_{\mathbb{A}}$

Similarity transformation of a tree BWA matrix

- Let T be an invertible matrix, then the following is a similarity transformation of a BWA matrix

$$\mathrm{Y} \overset{\triangle}{=} \mathrm{T}\mathcal{E}_{\mathbb{A}}\mathrm{T}^{-1}$$

- However, Y may <u>not</u> have the partitioned structure a BWA matrix
- The 1-resolvent matrix given by

$$(\mathbf{I} - \mathrm{Y})^{-1} = \mathrm{T}\mathbb{A}\mathrm{T}^{-1}$$

Permutation transformation of a tree BWA matrix

- A permutation is a square matrix such that

$$\mathsf{T}^{-1} = \mathsf{T}^*$$

- A similarity transformation via a permutation matrix retains the partitioned structure and is itself a BWA matrix with partitioned structure

$$\mathcal{E}_{\mathbb{B}} = \mathsf{T}\mathcal{E}_{\mathbb{A}}\mathsf{T}^*$$

- A permutation transformation of a BWA matrix corresponds to a reordering of the graph node indices, which as expected should not effect the BWA property.

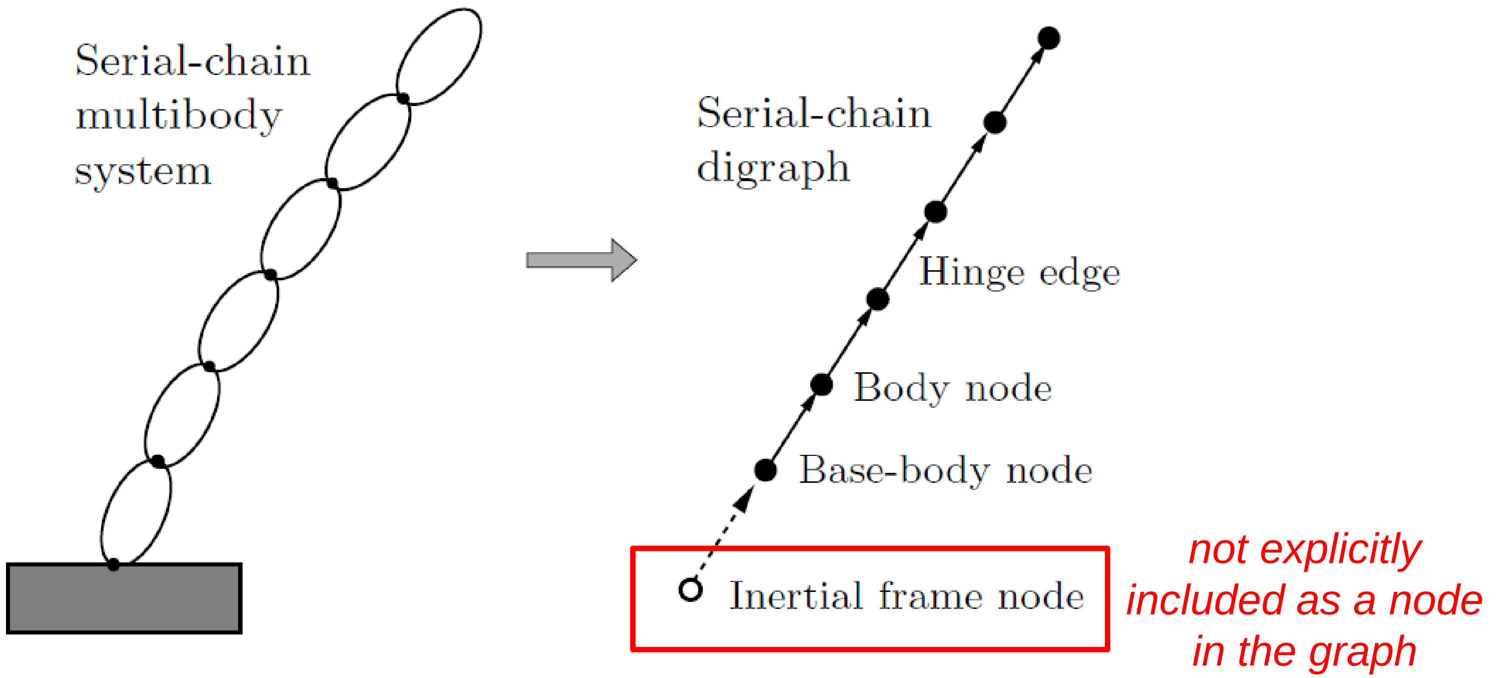
Recap of BWA development

- We have generalized the nilpotency properties and 1-resolvent existence using BWAs to
  - Tree systems with arbitrary branching structure
  - Arbitrary size trees
  - Block matrix elements
  - Non-square block matrix elements
- The next steps connect up the BWA development with multibody system analysis and SOA operators

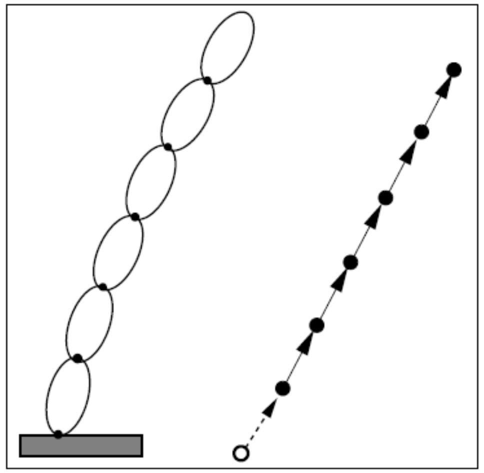
Multibody Systems

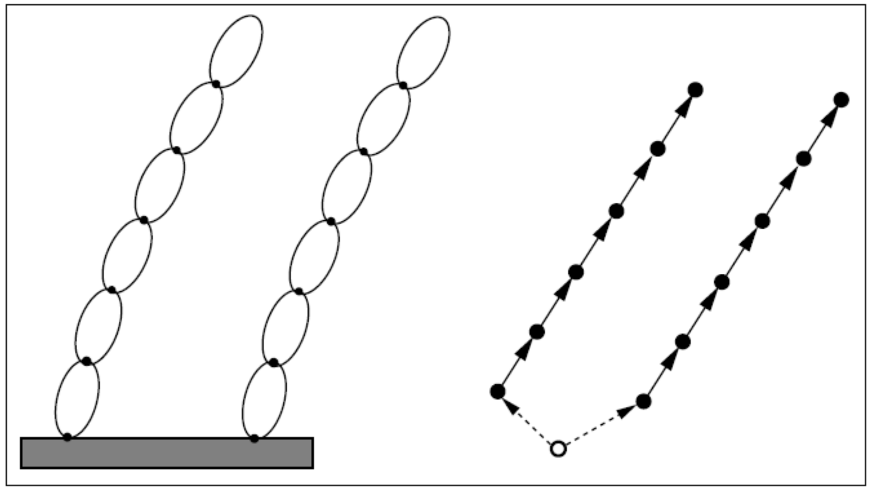
Graph representation for multibody systems

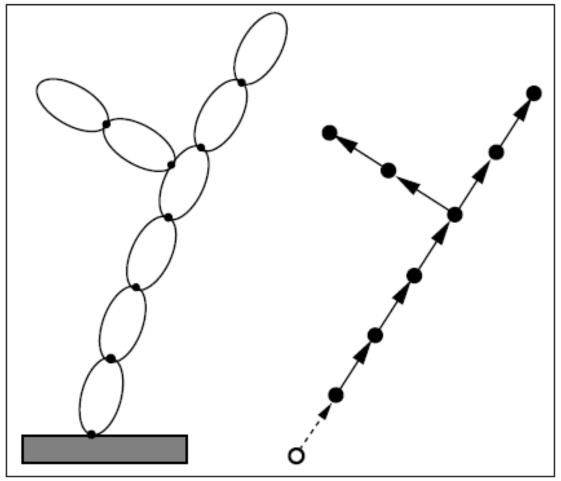
- Each body is a graph node, and each hinge a directed edge
- For a multibody system, have a digraph with n nodes and n edges

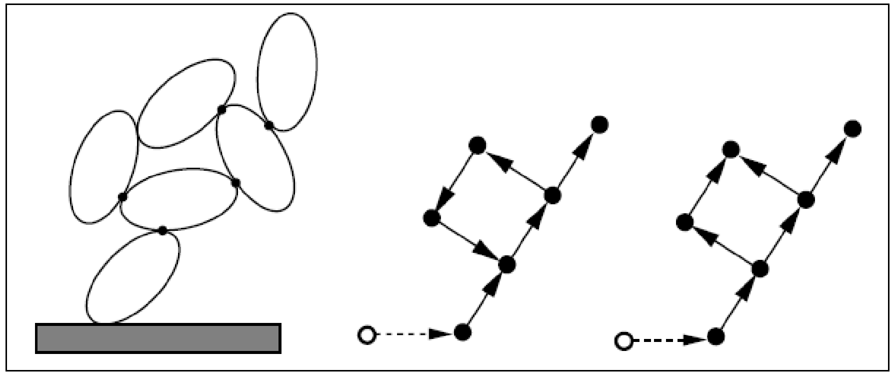

Serial-chain multibody system
Serial-chain digraph
Hinge edge
Body node
Base-body node
Inertial frame node
not explicitly included as a node in the graph

Examples of multibody graphs



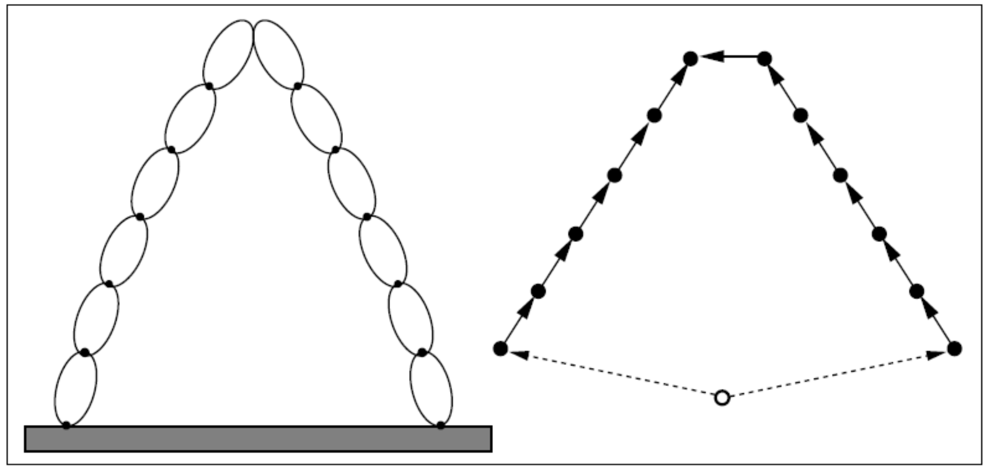


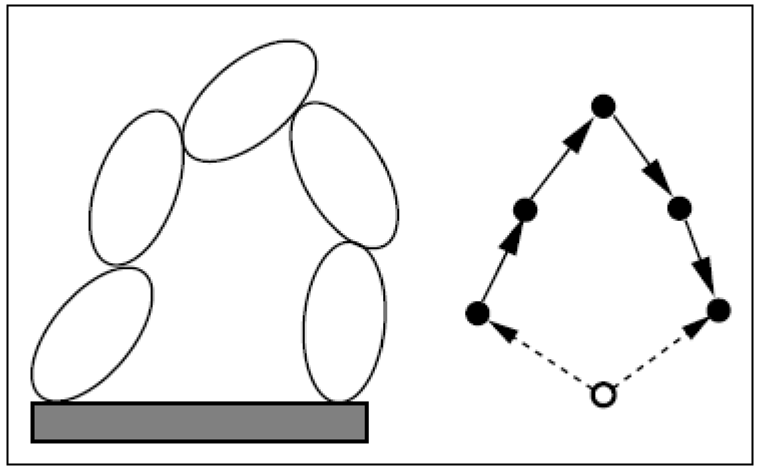




multiple graph options for loop systems

Examples of multibody graphs (contd)





Non-canonical serial chains

- For a canonical serial chain, the parent body for body k is body k+1, and have

$$\mathcal{V}(k) \overset{3.14,3.19a}{=} \phi^*(k+1, k)\mathcal{V}(k+1) + \Delta_{\mathcal{V}}(k)$$

$$\overset{3.7}{=} \phi^*(\boxed{k+1}, k)\mathcal{V}(k+1) + H^*(k)\beta(k)$$

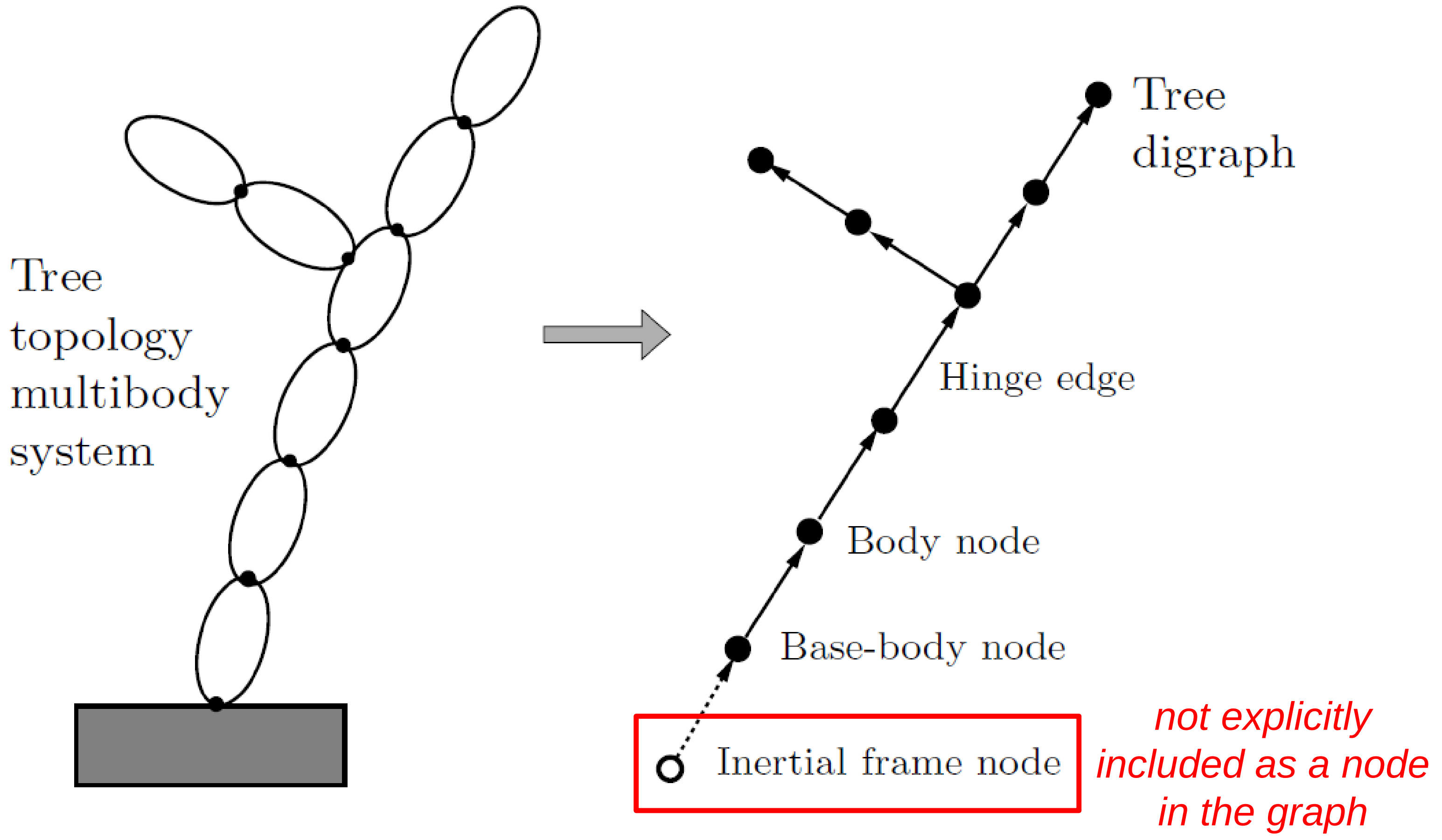
- Not so for non-canonical serial chains
  - Need to use the parent body designation

$$\mathcal{V}(k) = \phi^*(\boxed{\wp(k)}, k)\mathcal{V}(\wp(k)) + H^*(k)\dot{\theta}(k)$$

independent of
indexing convention

Tree Topology Multibody Systems

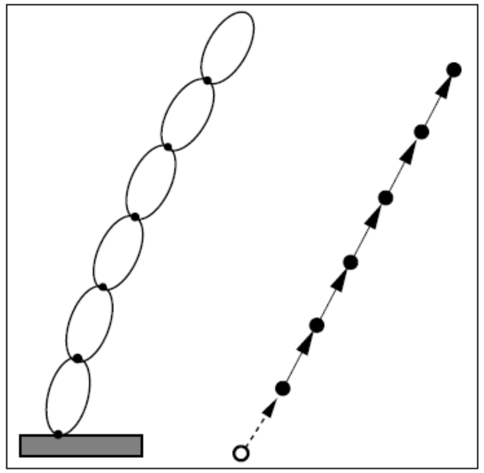
Graph for a tree multibody system



Serial-chain mbody/BWA matrix connection

Have seen that

$$\mathcal{E}_\phi \triangleq \begin{pmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \phi(2,1) & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \phi(3,2) & \dots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \phi(n,n-1) & \mathbf{0} \end{pmatrix}$$



is a serial-chain BWA matrix,

$$\boxed{\mathcal{E}_\phi(j,k) = \delta_{j,\wp(k)}\phi(\wp(k),k)}$$

with elements

$$w(\wp(k),k) = w(k+1,k) \triangleq \phi(k+1,k) = \phi(\wp(k),k)$$

Tree velocity kinematics

Body level velocity recurrence

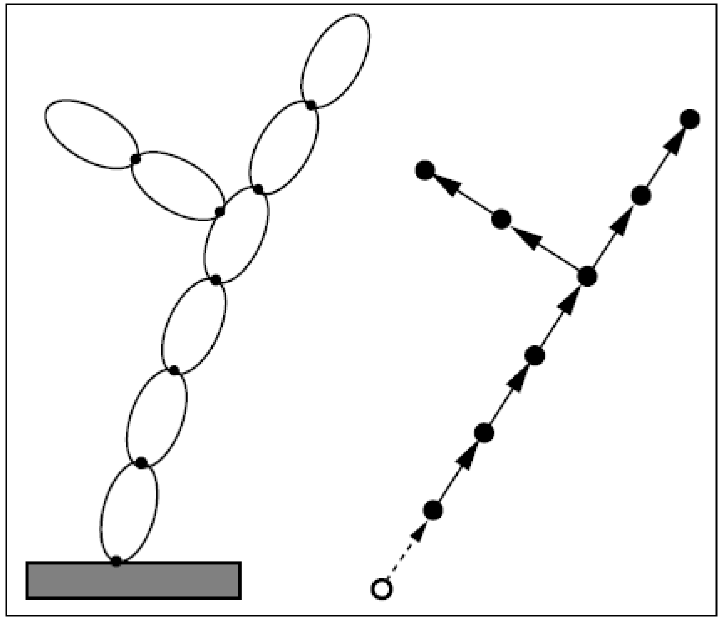
$$\mathcal{V}(k) = \phi^*(\wp(k), k)\mathcal{V}(\wp(k)) + H^*(k)\dot{\theta}(k)$$

With $\mathcal{E}_\phi(j, k) = \delta_{j,\wp(k)}\phi(\wp(k), k)$

we obtain the system-level expression

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

Once again $\mathcal{E}_\phi$ is a **tree BWA matrix**.



1-resolvent ϕ for tree mbody system

Can thus convert implicit expression

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

into explicit form

$$\mathcal{V} = \phi^{*}\mathrm{H}^{*}\dot{\theta}$$

using the tree BWA 1-resolvent

$$\boxed{\phi = (\mathbf{I} - \mathcal{E}_{\phi})^{-1}}$$

Body-level equations of motion

The kth body force balance expression is

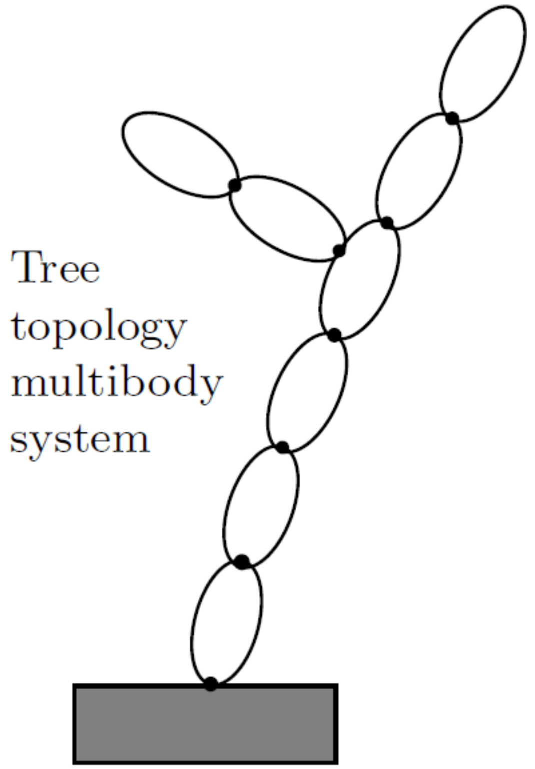
$$\mathfrak{f}(k) - \sum_{\forall j \in \mathcal{C}(k)} \phi(k,j)\mathfrak{f}(j) = \mathcal{M}(k)\alpha(k) + \mathfrak{b}(k)$$

or,

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

At the operator level,

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



Implicit to explicit

Use $\phi \overset{\triangle}{=} (\mathbf{I} - \mathcal{E}_\phi)^{-1}$ identity to convert implicit operator expressions into explicit ones

$$\boxed{\begin{aligned} \mathcal{V} &= \mathcal{E}_\phi^* \mathcal{V} + H^* \dot{\theta} \\ \alpha &= \mathcal{E}_\phi^* \alpha + H^* \ddot{\theta} + \mathfrak{a} \\ \mathfrak{f} &= \mathcal{E}_\phi \mathfrak{f} + \mathbf{M}\alpha + \mathfrak{b} \\ \mathcal{T} &= H\mathfrak{f} \end{aligned}} \Longrightarrow \boxed{\begin{aligned} \mathcal{V} &= \phi^* H^* \dot{\theta} \\ \alpha &= \phi^* (H^* \ddot{\theta} + \mathfrak{a}) \\ \mathfrak{f} &= \phi(\mathbf{M}\alpha + \mathfrak{b}) \\ \mathcal{T} &= H\mathfrak{f} \end{aligned}}$$

implicit expressions

explicit expressions

Jet Propulsion Laboratory
California Institute of Technology

System level equations of motion

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

$$\boxed{\begin{aligned} \mathcal{V} &= \phi^* H^* \dot{\theta} \\ \alpha &= \phi^* (H^* \ddot{\theta} + \mathfrak{a}) \\ \mathfrak{f} &= \phi(\mathbf{M}\alpha + \mathfrak{b}) \\ \mathcal{T} &= H\mathfrak{f} \end{aligned}} \Rightarrow \boxed{\begin{aligned} \mathcal{T} &= H\,\phi\,\left[\mathbf{M}\,\phi^*\,\left(H^*\,\ddot{\theta} + \mathfrak{a}\right) + \mathfrak{b}\right] \\ &= \mathcal{M}(\theta)\ddot{\theta} + \mathcal{C}(\theta, \dot{\theta}) \end{aligned}}$$

familiar **mass matrix** $\mathcal{M}(\theta) = H\phi\mathbf{M}\phi^* H^* \in \mathcal{R}^{\mathcal{N}\times\mathcal{N}}$

Newton-Euler factorization

Coriolis terms $\mathcal{C}(\theta, \dot{\theta}) \triangleq H\phi(\mathbf{M}\phi^*\mathfrak{a} + \mathfrak{b}) \in \mathcal{R}^{\mathcal{N}}$

Comments on tree equations of motion

$$\begin{aligned}\mathcal{T} &= \mathrm{H}\,\phi\,\left[\mathbf{M}\,\phi^*\,\left(\mathrm{H}^*\,\ddot{\theta} + \mathfrak{a}\right) + \mathfrak{b}\right] \\ &= \mathcal{M}(\theta)\ddot{\theta} + \mathcal{C}(\theta, \dot{\theta})\end{aligned}$$

- The equations of motion are identical in form at the operator level to the serial-chain equations of motion!
  - These hold for arbitrary system size and branching
- The differences are
  - At the component level we now have to work with multiple children bodies
  - The operator structure is different – however they are both tree BWA matrices

Comments

- We have been able to generalize the notion of the $\mathcal{E}_\phi$ and ϕ operators & **structure** from serial chains to trees
  - The crucial step was to look at the general property of BWA matrices associated with graphs and to recognize that $\mathcal{E}_\phi$ is a tree BWA matrix
- From the tree BWA property alone we could
  - Establish the existence of the 1-resolvent
  - Establish sparsity property based on topological structure
  - Establish disjointedness property of the 1-resolvent and its transpose
  - Establish the chain rule property for the elements of the 1-resolvent
- We did not require canonical indexing, and triangularity assumptions at all!

Comments (contd)

- We have earlier used a canonical serial chain to develop the SOA operator analysis and algorithm
  - The simple structure of canonical serial-chains allowed us to build up the techniques as well as our intuition
- But as we are starting to see, neither the serial-chain, nor the canonical nature are really that important
  - It is the BWA part that matters
- The specific block entries of the BWA matrix did not matter either
  - In fact they do not even have to be *square* or *rigid body transformation matrices!*

Recap

Summary

- Used graph theory analytical concepts to define the notion of BWA matrices for trees and graphs
- Showed that tree BWA matrices have a well defined 1-resolvent matrix
- Showed that the $\mathcal{E}_{\phi}$ spatial operator for serial chains is a tree BWA matrix
- Developed equations of motion for a tree system using tree BWA operators
- The spatial operator expressions remain unchanged from serial to tree systems

SOA Generalization Track Topics

8. **Graph theory based structure** – BWA matrices, connection to multibody systems
9. **Tree topology systems** – generalization to tree topology rigid body systems, SKO/SPO operators, gather/scatter algorithms
10. **Closed-chain dynamics (cut-joint)** – holonomic and non-holonomic constraints, cut-joint method, operational space inertia, projected dynamics
11. **Closed-chain dynamics (constraint embedding)** – constraint embedding for graph transformation, minimal coordinate closed-chain dynamics
12. **Flexible body dynamics** – Extension to flexible bodies, modal representations, recursive flexible body dynamics