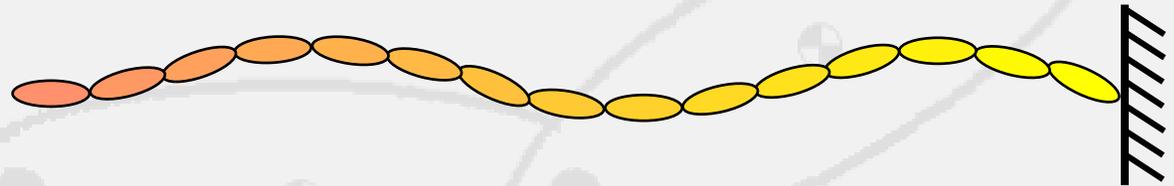




Jet Propulsion Laboratory
California Institute of Technology

PyCraft

***A Unifying Multibody Dynamics Algorithm
Development Workbench***



John Ziegler RPI '06

Mentor: Dr. Abhinandan Jain

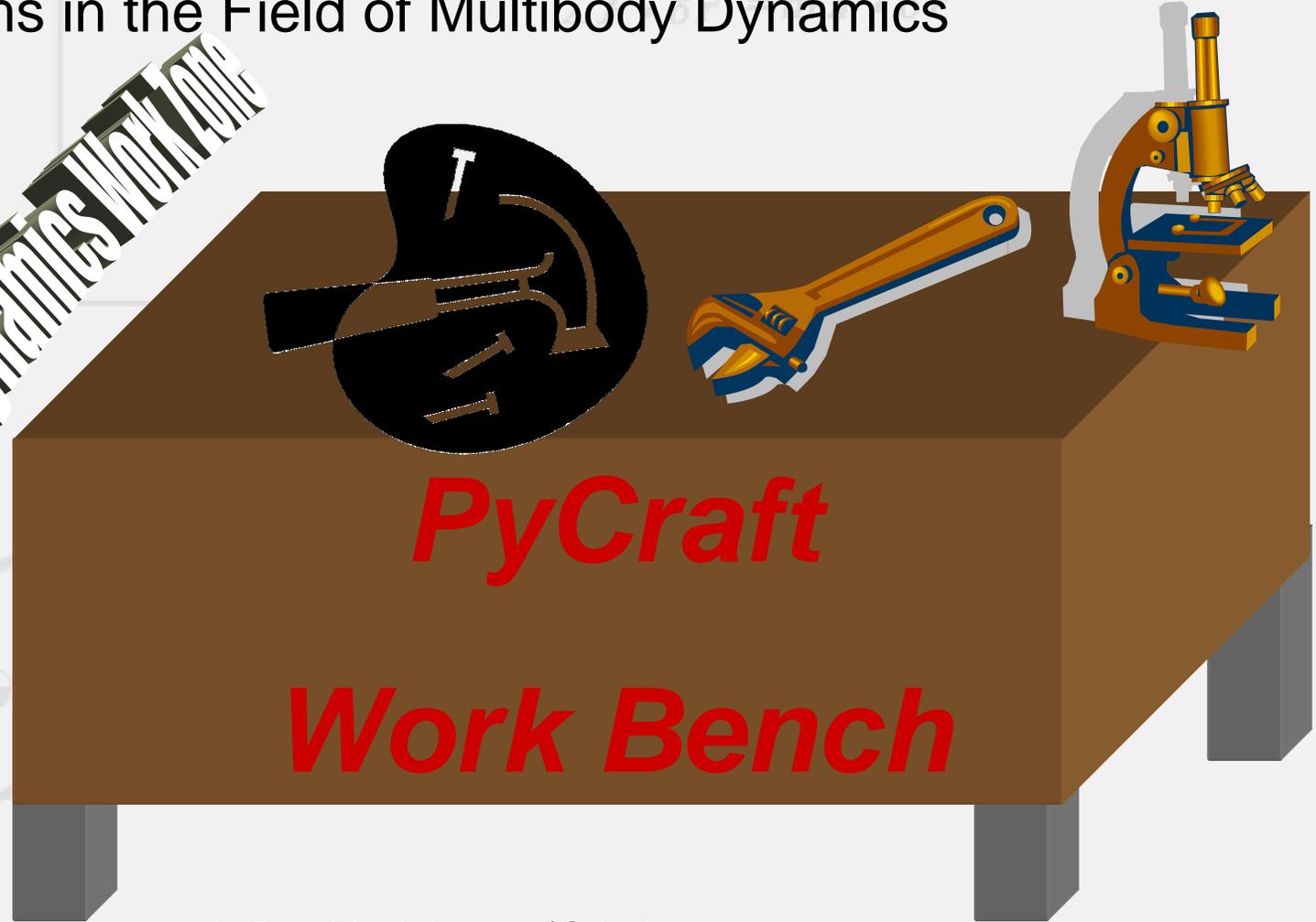
Aug 18, 2005



A Work Bench for Multibody Body Dynamics Algorithms

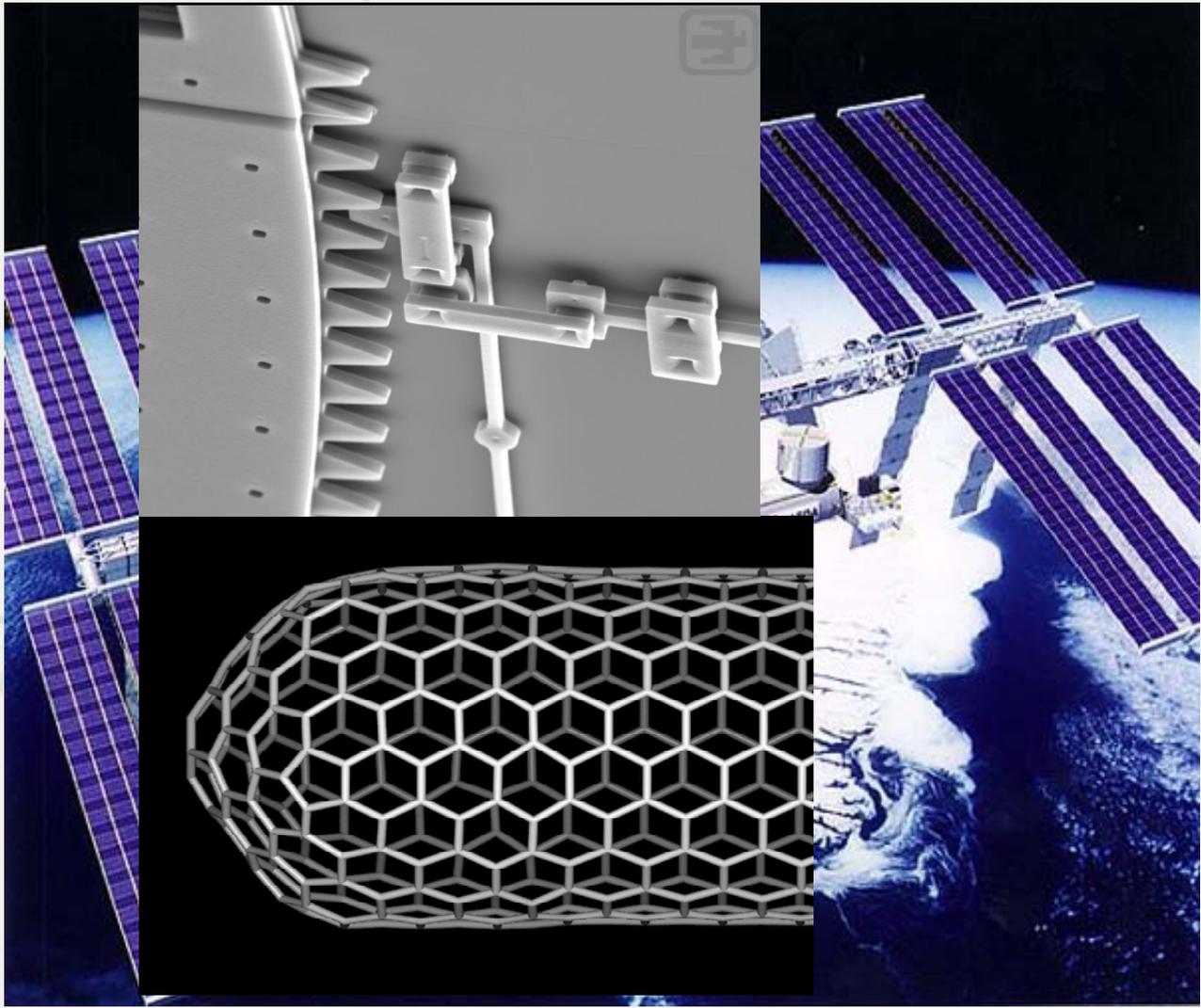
- With our efforts, this module will serve as a development tool for future modeling and simulation algorithms in the Field of Multibody Dynamics

Multibody Dynamics Work Bench





Example Multibody Systems





Dynamics

- Newton's 3 Laws

- $F = Ma$
- Conservation of Momentum
- Equal and opposite Forces

$$\Sigma \vec{F} \cdot \delta \vec{r} = m \vec{a} \cdot \delta \vec{r}$$

- Modeling, Simulation, and Control of the motion of Physical Systems

- Multibody Dynamics

- Extremely Hard Research Field
- Challenging Mathematics,
- Thousands of Equations & Symbols, Constraints, Coupling, Flexible Bodies, Information Overload
- **Almost impossible** to program

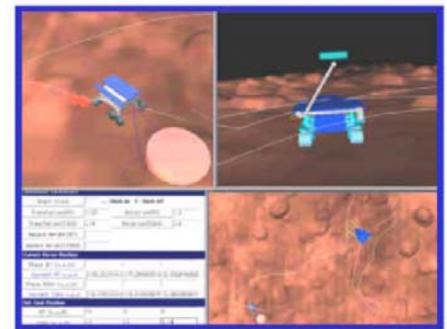


Dynamics and Real Time Simulation Laboratory: Software Development

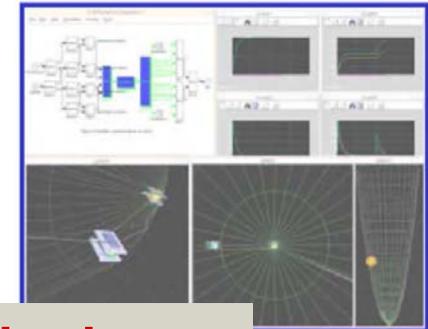
DSENGS
Entry, Descent & Landing Simulator



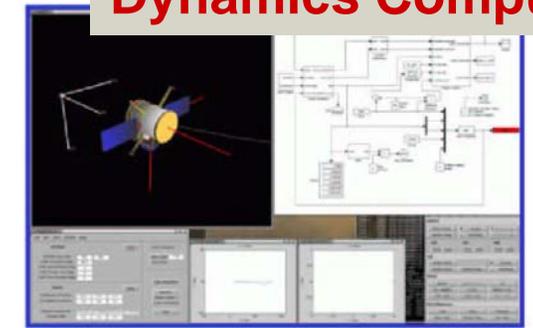
ROAMS
Planetary Rover Simulator



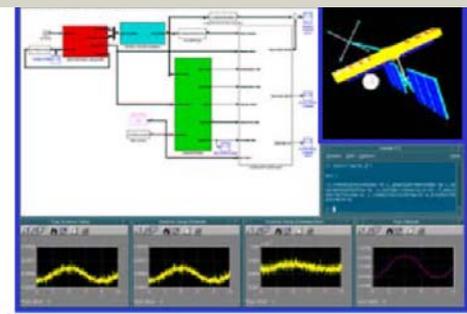
Formation Flying
Distributed Real-Time Simulations (Starlight)



Run by High Fidelity, Optimized Multibody Dynamics Computational Software



Rendezvous & Sample Capture
(Mars RSC, ST-6, CNES'07)



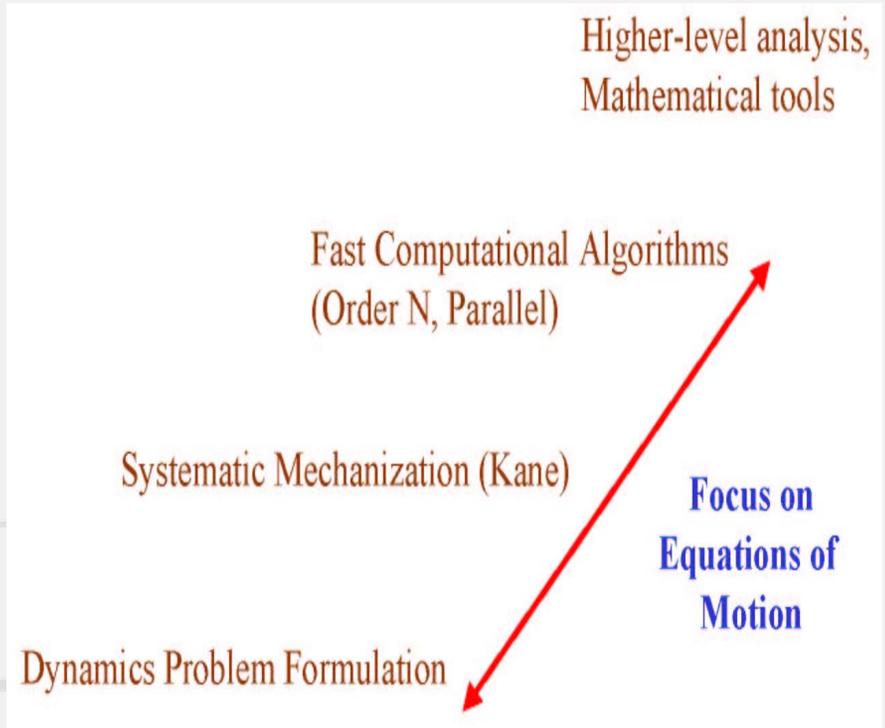
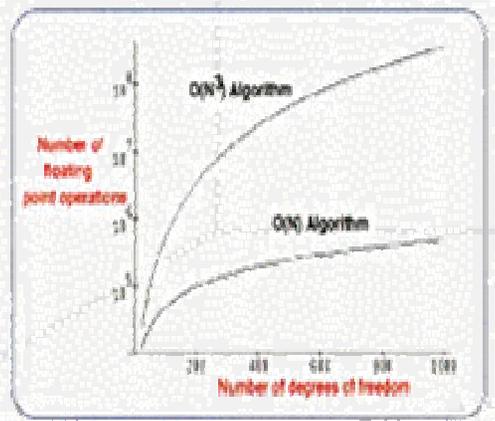
Flexible Interferometers
Integrated Modeling (SIM)



Dynamics Software

The DARTS Computational Algorithm

Uses $O(N)$ Spatial Operator Algebra computational algorithms



- Utilize Dynamics software to model, simulate, and control multibody systems
- Utilize Simplifying Mathematical tools, utilize algorithms, recursions
- **New Problem, (This is where my role begins)**
- **Different software for different applications, hard to develop new mathematics, algorithms, optimize, etc...**

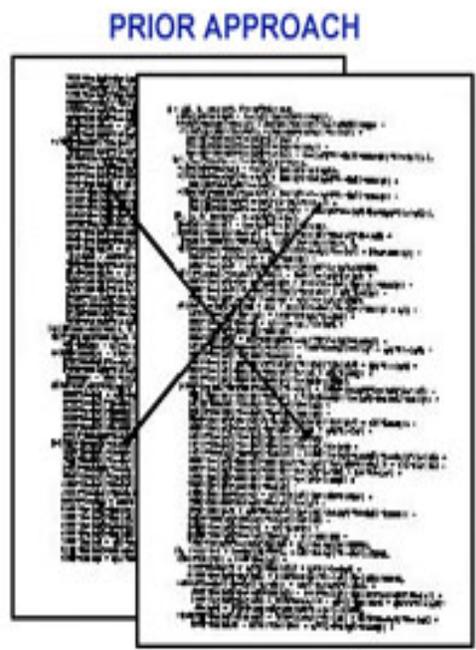


Utilize Mathematical Tools to Reduce Complexity

Spatial Operator Algebra

$$J = B^* \Phi^* H^*$$

Build Operators



$$F = Ma$$

- $F_1 + F_2 + \dots F_n = m_1 * a_1$
- $F_1 + F_2 + \dots F_n = m_2 * a_2$
- $F_1 + F_2 + \dots F_n = m_3 * a_3$
- $F_1 + F_2 + \dots F_n = m_4 * a_4$
- $F_1 + F_2 + \dots F_n = m_5 * a_5$
- $F_1 + F_2 + \dots F_n = m_6 * a_6$
- $F_1 + F_2 + \dots F_n = m_7 * a_7$
- $F_1 + F_2 + \dots F_n = m_8 * a_8$
- $F_1 + F_2 + \dots F_n = m_9 * a_9$
- $F_1 + F_2 + \dots F_n = m_{10} * a_{10}$
- $F_1 + F_2 + \dots F_n = m_{11} * a_{11}$
- $F_1 + F_2 + \dots F_n = m_{12} * a_{12}$
- $F_1 + F_2 + \dots F_n = m_{13} * a_{13}$
- $F_1 + F_2 + \dots F_n = m_{14} * a_{14}$
- $F_1 + F_2 + \dots F_n = m_{15} * a_{15}$
- $F_1 + F_2 + \dots F_n = m_{16} * a_{16}$
- $F_1 + F_2 + \dots F_n = m_{17} * a_{17}$
- $F_1 + F_2 + \dots F_n = m_{18} * a_{18}$

Equations of Motion

$$M(\theta)\ddot{\theta} + C(\theta, \dot{\theta}) = T$$

N x N size Matrix

N x 1 Vectors

*** OR**



Operators Reduce Complexity to Single Equations

Equations of Motion

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

θ = generalized hinge coordinates

T = generalized forces

$\mathcal{M}(\theta)$ = mass matrix

$\mathcal{C}(\theta, \dot{\theta})$ = velocity-dependent Coriolis/centrifugal forces

$V = \phi^* H^* \dot{\theta}$ spatial velocities

$\alpha = \phi^* [H^* \ddot{\theta} + a]$ spatial accelerations

$f = \phi [M\alpha + b]$ inter-body spatial forces

$T = Hf$ generalized forces

$$T = \underbrace{H\phi M\phi^* H^*}_{\mathcal{M}(\theta)} \ddot{\theta} + \underbrace{H\phi [M\phi^* a + b]}_{\mathcal{C}(\theta, \dot{\theta})}$$

Each Operator is now a Matrix!

$$\sum F \cdot \delta \bar{r} = m \bar{a} \cdot \delta \bar{r}$$

Spatial Level

$$V = \text{col} \{ V(k) \} = \phi^* H^* \dot{\theta}$$

Operator Expression

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

Component Level

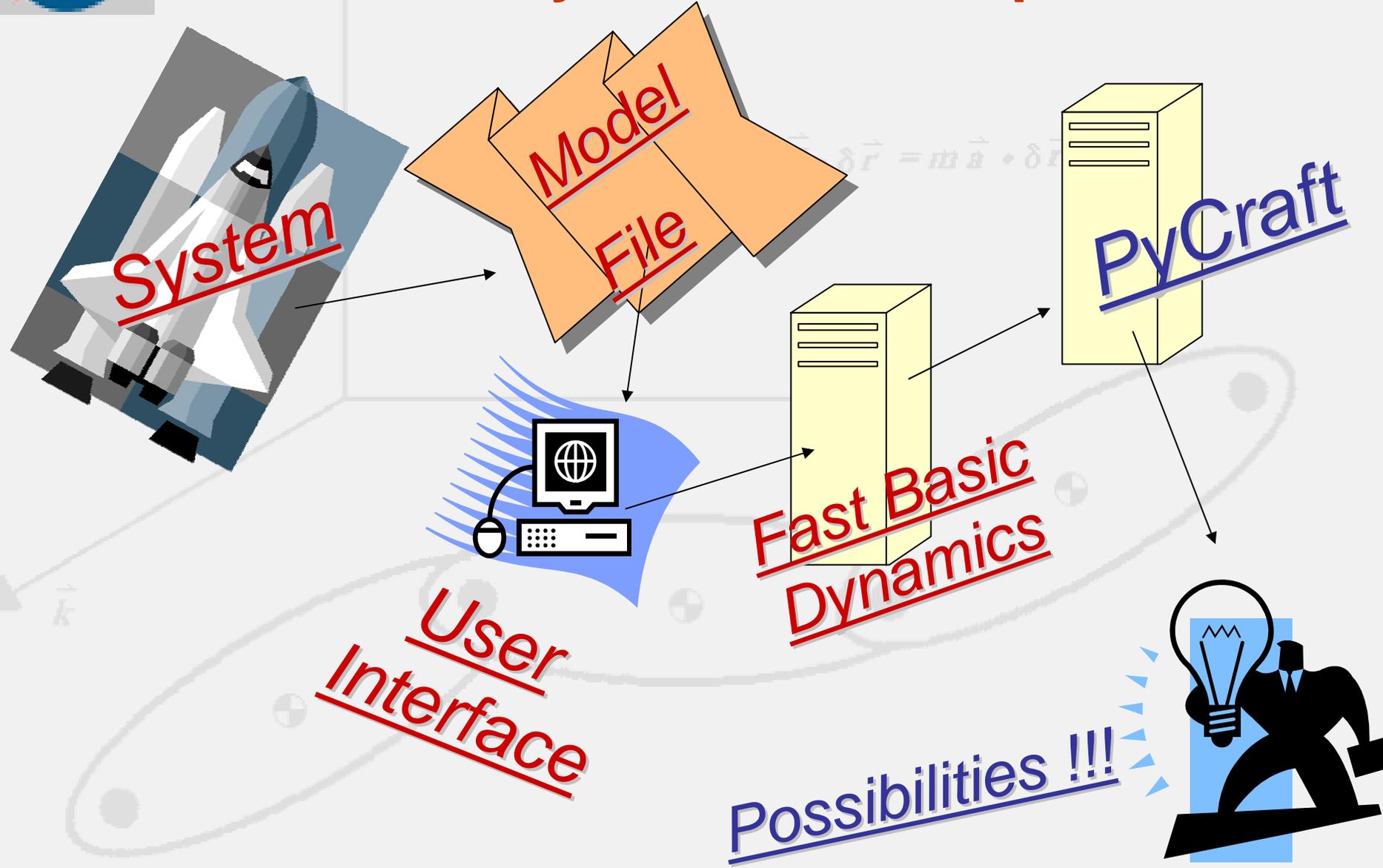
$$\Leftrightarrow \begin{cases} V(n+1) = 0 \\ \text{for } k = N \dots 1 \\ V(k) = \phi^*(k+1, k) V(k+1) + H^*(k) \dot{\theta}(k) \\ \text{end loop} \end{cases}$$

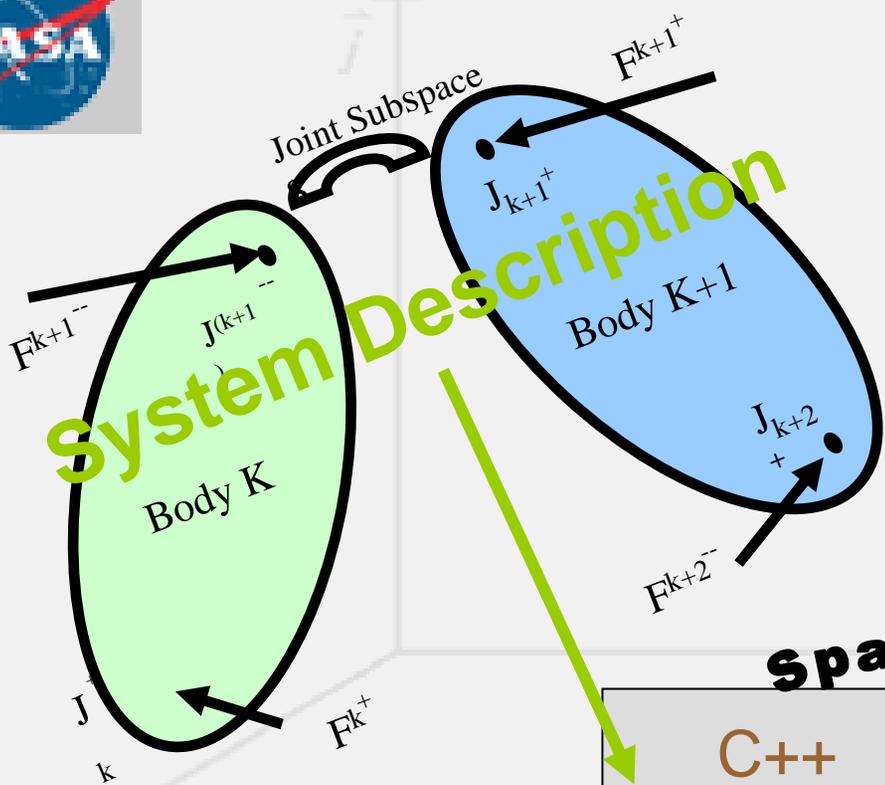
Computational Algorithm

$$H = \begin{pmatrix} H(1) & 0 & \dots & 0 \\ 0 & H(2) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & H(n) \end{pmatrix}$$



The PyCraft Road Map





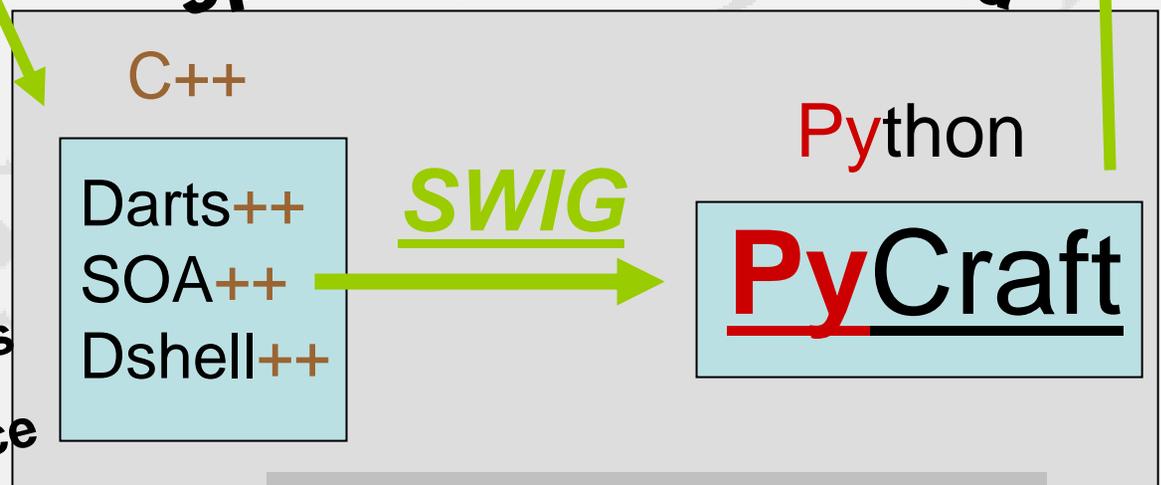
System Description

Dynamics Applications
Future Algorithm Development

$$\Sigma \vec{F} \cdot \delta \vec{r} = m \vec{a}$$

Spatial Operator Algebra

Optimized Basic Dynamics
Computations
User Interface

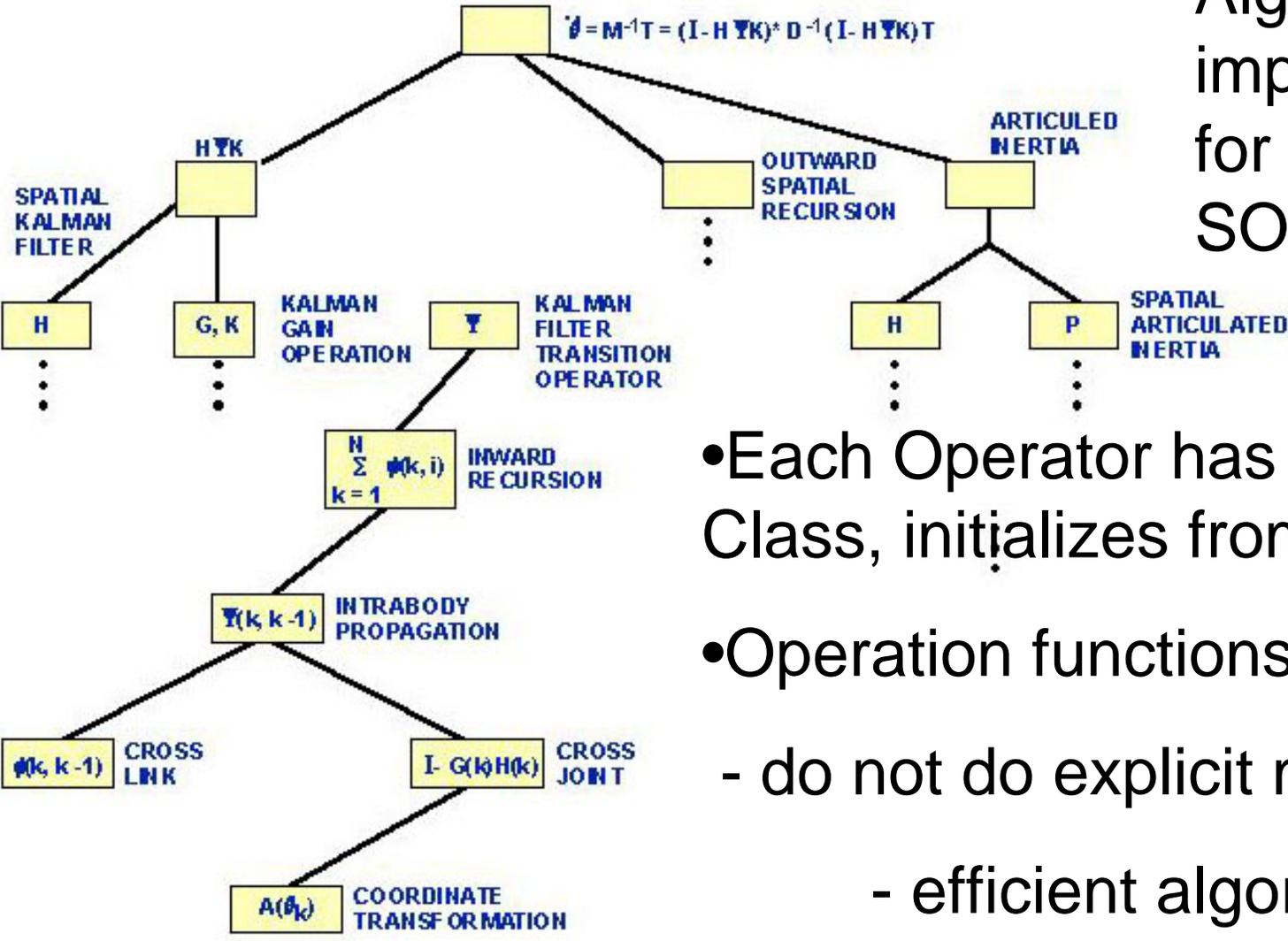


The Total Software Package



SOA Operators with Object Orientated Programming

Algorithms implemented for Building SOA Objects



- Each Operator has its own Class, initializes from model file
- Operation functions * , $+$, $-$,
 - do not do explicit math
 - efficient algorithms used



PyCraft Make MBS Easy

MBS User Interface -Just like Matlab!

```
>>> from Math import SOA_Py
>>> execfile('model.py')
>>> scObj =
    DshellObj.spacecraftObj('DefaultSC')
>>> mbodyObj = scObj.mbodyObj('darts')
>>> mbodyObj.evalSensorKinematics()
>>> mbodyObj.evalActuatorKinematics()
>>> from PyCraft import PyCraft
>>> Phi = PyCraft.Phi(mbodyObj)
>>> PhiStar = PyCraft.PhiStar()
>>> Thetadot =
    PyCraft.Thetadot(mbodyObj)
>>> H = PyCraft.H()
>>> HStar = PyCraft.HStar()
>>> M = PyCraft.M()
```

```
>>> V = PhiStar*HStar*Thetadot
```

```
>>> BStar = PyCraft.BStar(mbodyObj, 0)
```

```
>>> Jacobian = BStar*(PhiStar*HStar)
```

```
>>> alphasip = Jacobian*Thetadoubledot
```

```
>>> print pformat(alphasip())
```

```
(-0.007081204690359221, -0.013761772880363957,
 0.00022901925837621328, -43.446976936483701, -
 8.7413024772829448, 8.9785110488587563)
```

```
>>> T = H*Phi*M*PhiStar*HStar*Betadot
```

```
>>> Betadot = ((Spatial_I-
  H*Psi*K)._transpose())*D_Inverse*(Spatial_I-H*Psi*K)*T
```

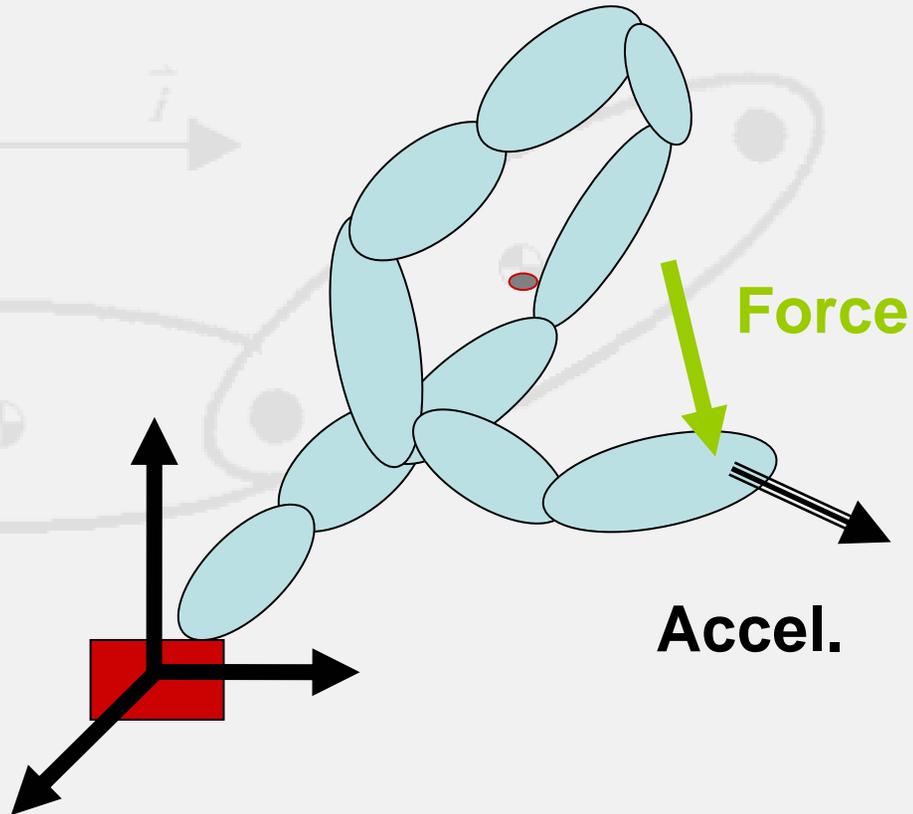


PyCraft Applications

- Inverse Dynamics
 - Input accel, output Forces
- Forward Dynamics
 - Input Forces, output accel
- Diverse Systems
 - Tree, Closed Chain, + Flexible Systems
- Higher Level Research Areas
 - Sensitivity Analysis
 - Operational Space Inertia
 - Linearized & Diagonalized Dynamics for Control
 - Parameter Optimization

$$T = \underbrace{H\phi M \phi^* H^*}_{\mathcal{M}(\theta)} \ddot{\theta} + \underbrace{H\phi [M \phi^* a + b]}_{c(\theta, \dot{\theta})}$$

$$\begin{aligned} \Sigma \vec{F} \cdot \delta \vec{r} &= m \vec{a} \cdot \delta \vec{r} \\ \ddot{\theta} &= \mathcal{M}^{-1}(\theta) [T - c(\theta, \dot{\theta})] \\ &= [I - H\psi K]^* D^{-1} [I - H\psi K] [T - c(\theta, \dot{\theta})] \end{aligned}$$





Conclusions

- The Mathematical tool called Spatial Operator Algebra is used to model of the complexity in MBS dynamics
- SOA is hard implement, reusing dynamics software at JPL DARTS Lab, and coding the PyCraft Module makes these easier.
- PyCraft, with its symbolic and innate algorithmic nature, naturally leads the derivation of new algorithms and ways of solving dynamics problems

Questions?



Unified Framework for Deriving Families of Algorithms

$\mathcal{M} = H\phi M\phi^* H^*$ → $O(\mathcal{N})$ Newton-Euler Inverse Dynamics
 $O(\mathcal{N}^2)$ Composite Body Algorithm for \mathcal{M}
 $O(\mathcal{N}^3)$ Forward Dynamics

$\mathcal{M} = [I + H\phi K]D[I + H\phi K]^*$ → $O(\mathcal{N}^2)$ Forward Dynamics

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

$\mathcal{M}^{-1} = [I - H\psi K]^* D^{-1} [I - H\psi K]$ → $O(\mathcal{N})$ Articulated Body Forward Dynamics
 $O(\mathcal{N}^2)$ Computation of \mathcal{M}^{-1}

The spatial operator approach provides a unified dynamics formulation for serial/tree, rigid/flexible multibody systems.



Inverse Dynamics Solutions

$$\begin{aligned} V &= \phi^* H^* \dot{\theta} && \text{spatial velocities} \\ \alpha &= \phi^* [H^* \ddot{\theta} + a] && \text{spatial accelerations} \\ f &= \phi [M \alpha + b] && \text{inter-body spatial forces} \\ T &= H f && \text{generalized forces} \end{aligned}$$

$$T = \underbrace{H \phi M \phi^* H^*}_{\mathcal{M}(\theta)} \ddot{\theta} + \underbrace{H \phi [M \phi^* a + b]}_{c(\theta, \dot{\theta})}$$

This factorization of \mathcal{M} is called the *Newton–Euler Factorization of the Mass Matrix*.



Forward Dynamics Solution

```

P+(0) = 0
for k = 1...N
  P(k) = phi(k, k-1)P+(k-1)phi*(k, k-1) + M(k)
  D(k) = H(k)P(k)H*(k)
  G(k) = P(k)H*(k)D^-1(k)
  tau_bar(k) = I - G(k)H(k)
  P+(k) = tau_bar(k)P(k)
  psi(k+1, k) = phi(k+1, k)tau_bar(k)
end loop

```

- These operator expressions directly map into a sequence of recursive computational algorithms which reduce computational cost of solving the equations of motion of $O(n^3)$ to $O(n)$
- A Riccati equation for articulated body inertia needs to be solved to obtain innovations representation

$$\ddot{\theta} = \mathcal{M}^{-1}(\theta) [T - \mathcal{C}(\theta, \dot{\theta})]$$

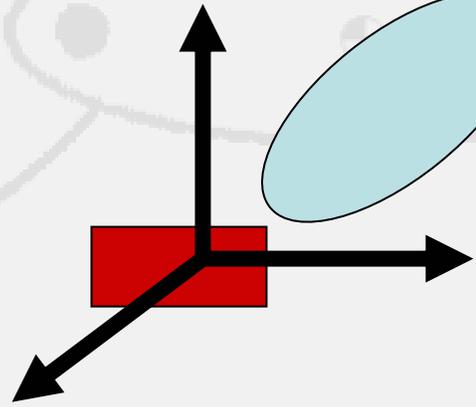
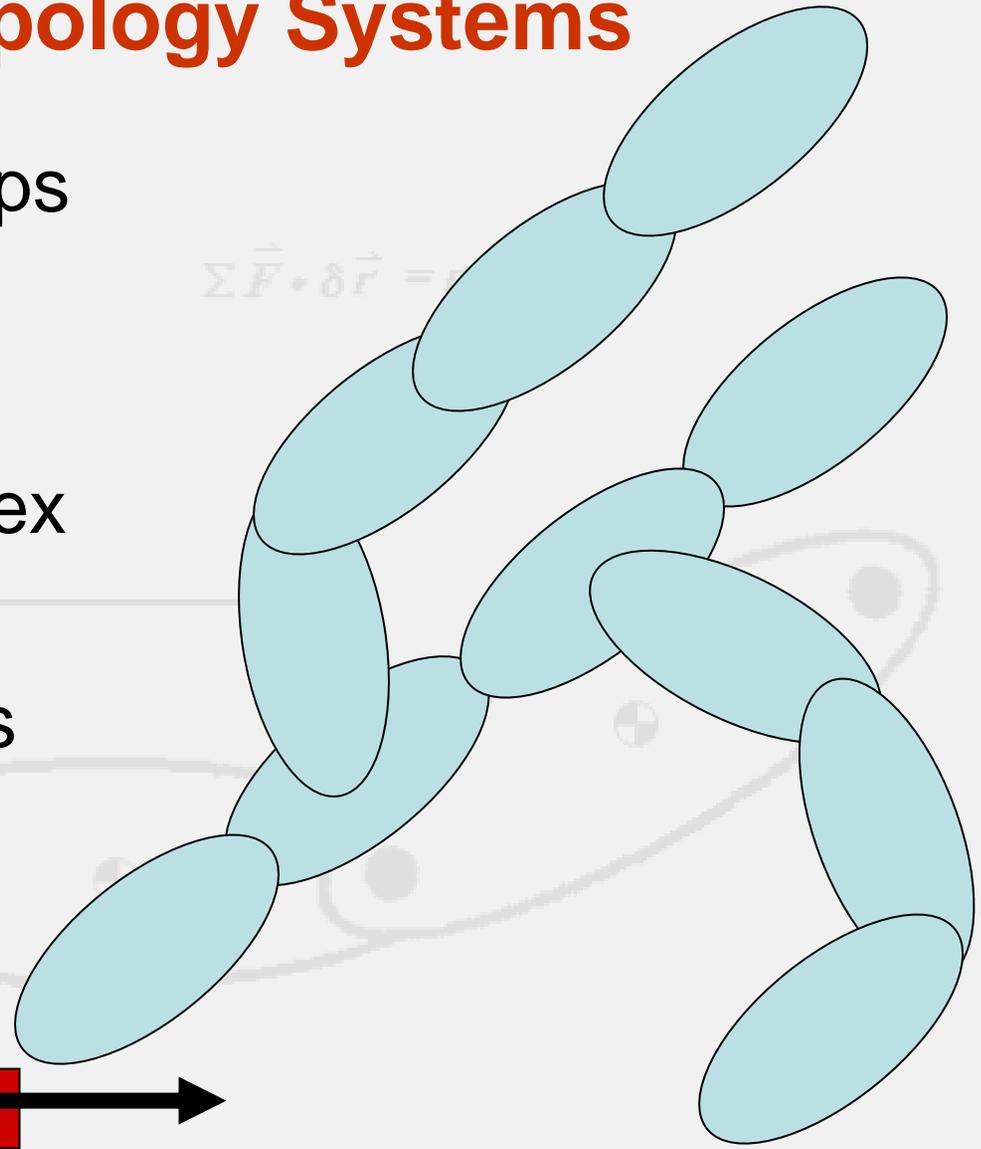
$$= [I - H\psi K]^* D^{-1} [I - H\psi K] [T - \mathcal{C}(\theta, \dot{\theta})]$$



Tree Topology Systems

- Construct Index Maps
- Branch index
- Body on branch index
- Global body index
- Predecessor checks

$$\sum \vec{F} \cdot \delta \vec{r} = 0$$





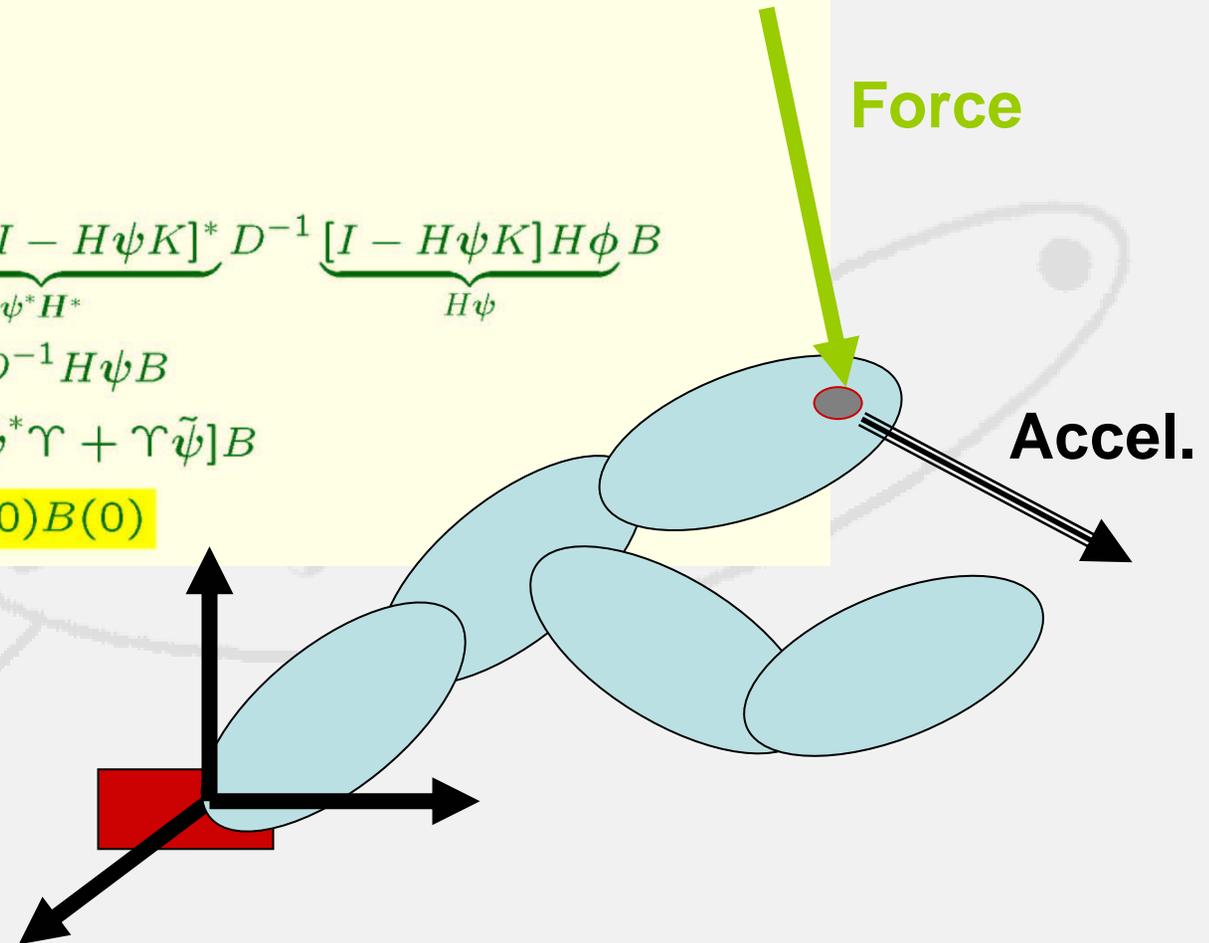
Operational Space Inertia

The Operational Space Mass Matrix Λ is the joint space mass matrix \mathcal{M} reflected to the end-effector.

$$\begin{aligned}
\Lambda^{-1} &= J\mathcal{M}^{-1}J^* \\
&= B^* \underbrace{\phi^* H^* [I - H\psi K]^*}_{\psi^* H^*} D^{-1} \underbrace{[I - H\psi K] H \phi}_{H\psi} B \\
&= B^* \psi^* H^* D^{-1} H \psi B \\
&= B^* [\Upsilon + \tilde{\psi}^* \Upsilon + \Upsilon \tilde{\psi}] B \\
&= B^*(0) \Upsilon(0) B(0)
\end{aligned}$$

Force

Accel.



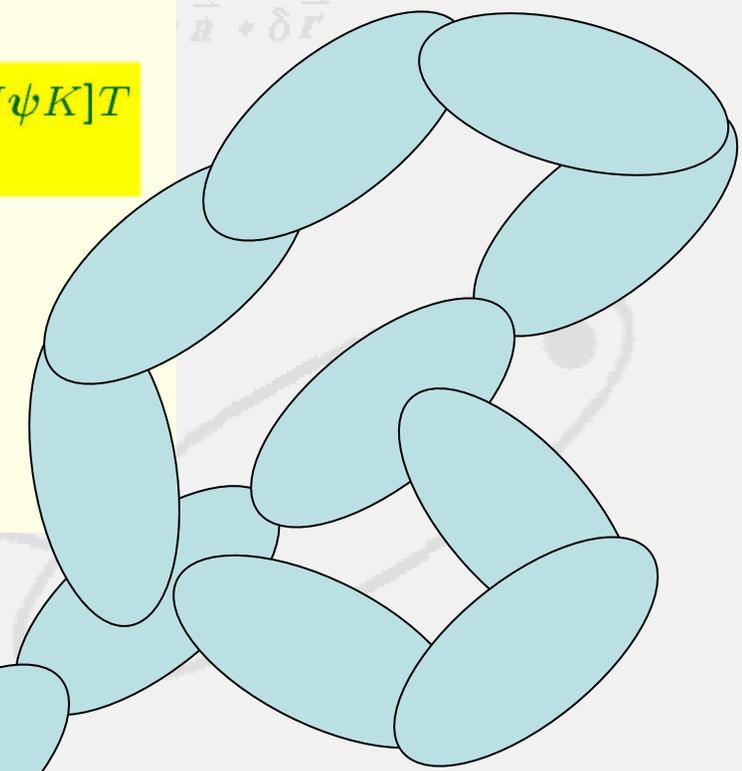


Closed Chain & Constrained System Dynamics

Closed-loop systems involve closure constraints.

$$\ddot{\theta} = [I - H\psi K]^* D^{-\frac{1}{2}} \underbrace{[I - B^* \Omega^{-1} B]}_{\text{extra term}} D^{-\frac{1}{2}} [I - H\psi K] T$$

1. solve "free" dynamics
2. solve for constraint forces
3. correct free dynamics solution





Sensitivity Analysis

- Derivatives of Spatial Operators with respect to degrees of freedom
- Leads to Time Derivates of Spatial Operators
- Applications:
 - Selective Optimization of Parameters
 - Coriolis term calculation
 - Diagonalized + Linearized Dynamics
 - Efficient Control algorithms



Operators as Mathematical Tools

Operator Factorization & Mass Matrix Inversion

Spatial Operator Identities

Newton-Euler Factorization of \mathcal{M}

$\mathcal{M} = H\phi M\phi^* H^*$ Non-square factors

Innovations Factorization of \mathcal{M}

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

Analytic inverse of $[I + H\phi K]$

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

Operator Factorization of \mathcal{M}^{-1}

$\mathcal{M}^{-1} = [I - H\psi K]^* D^{-1} [I - H\psi K]$ LDU decomposition of \mathcal{M}^{-1}

$$\tau P \tau^* = \tau P$$

$$\psi^{-1} - \phi^{-1} = KH$$

$$[I - H\psi K]H\phi = H\psi$$

$$\phi K [I - H\psi K] = \psi K$$

$$\phi M \phi^* = R + \tilde{\phi} R + R \tilde{\phi}^*$$

$$\psi M \psi^* = P + \tilde{\psi} P + P \tilde{\psi}^*$$

$$H\psi M \psi^* H^* = D$$

$$\psi^* H^* D^{-1} H\psi = \Upsilon + \tilde{\psi}^* \Upsilon + \Upsilon \tilde{\psi}$$

• These expressions reflect the deep structure of the key dynamics quantity - the mass matrix, which the spatial operator techniques are able to exploit to obtain closed-form expressions for its inverse.



Applications / Extensions

Generalizations

- Inverse Dynamics
- Forward Dynamics
- Mass Matrix
- Inverse kinematics
- Tree-topology systems
- Closed-chain
- Flexible multibody systems
- Geared dynamics
- Prescribed motion

Extensions

- Mass Matrix Inverse
- Mass Matrix determinant
- Operational Space Mass Matrix
- Dynamics with Joint Flexibility
- Under-Actuated Dynamics
- Generalized Jacobian
- Disturbance Jacobian
- Linearized Dynamics Models
- Base-invariant dynamics
- Sensitivity Computations
- Diagonalized Dynamics

Applications

- Space system simulations
 - Closed-loop simulations
 - Hardware-in-the-loop
 - Flexible s/c models
 - GN&C/Matlab simulations
 - Engineering simulations
 - Monte Carlo simulations
 - System design studies
- Space Mission Domains
 - Cruise/Orbiter spacecraft
 - Large Sciencecraft platforms
 - Formation flying s/c
 - Entry, descent and landing
 - Rendezvous and sample capture
 - Surface planetary rovers
- Molecular dynamics
 - Internal coordinate
 - Nose-Hoover dynamics
 - Fixman potential



Conclusions

- The Spatial Operator Algebra builds upon mathematical parallels between Kalman filtering theory and multibody dynamics.
- The operators are natural mathematical tools for developing new analytical insights into system dynamics.
- There is a natural mapping between operator expressions and efficient computational algorithms.
- Described several applications including $O(\mathcal{N})$ algorithms, diagonalization, compensating mass matrix potential, mass matrix sensitivities etc.



Future Work

- Testing of closed chain, constraints, and sensitivity
- Base invariant dynamics
- Diagonalized and linearized dynamics + control
- Extension to Flexible dynamics
- Prescribed motion dynamics

$$\Sigma \vec{F} \cdot \delta \vec{r} = m \vec{a} \cdot \delta \vec{r}$$