SPART: AN OPEN-SOURCE SIMULATOR FOR SPACECRAFT ROBOTIC ARM DYNAMIC MODELING AND CONTROL

Dr. Josep Virgili Llop, NPS-Mechanical and Aerospace Engineering
CPT. Jerry V. Drew II, NPS-Mechanical and Aerospace Engineering
Prof. Marcello Romano, NPS-Mechanical and Aerospace Engineering
Outline

• Motivation

• Other existing tools

• DeNOC kinematic and dynamic modelling approach

• Application examples
Motivation

Requirements:

• For teaching and research alike.

• Need to be able to integrate with the existing toolchain for the experimental test bed (Simulink RTW with an RTAI Linux target).

• Students are mainly familiar with Matlab/Simulink.

• Access into the kinematic and dynamic properties is required.

• Collection of function as building blocks for rapid prototyping of guidance and control algorithms (flexible and modular).
Motivation

Naval Postgraduate School - Floating Spacecraft Simulator

- Floating Spacecraft Simulator
- Motion Capture Cameras
- Modular Robotic Arm
- Granite Monolith

Floating Spacecraft Simulator

Dr. Virgili – jvirgili@nps.edu
Naval Postgraduate School - Floating Spacecraft Simulator
Other open source tools

Other existing tools:
- ODE
- Bullet
- DART
- Simbody

General physics engines (focused on efficiency), written in C/C++, generally without access to kinematics and dynamic properties and no repository of control algorithms.

Other frameworks
- ROS
- GAZEBO
- MoveIt!

Could provide complement the SPART capabilities (i.e. visualization, navigation and networking) and thus SPART will
SPART features:

• Toolkit – Not a generic physics engine

• Based on MATLAB/Simulink for rapid algorithm prototyping and easy integration with NPS-FSS development toolchain.

• Repository of control algorithms.

• Repository of analysis tools.

• Automatic Code Generation is used to generate efficient C/C++ that can run on embedded hardware.
Kinematics

$q \leftrightarrow$ Joint space variables
- $q_0$ base-spacecraft state
- $q_m$ manipulator joint states

Coordinate transformations can be obtained recursively

$$\mathcal{T}_i \mathcal{J}_i = \mathcal{T}_0 \mathcal{L}_0 T_{J_{0+1}} \prod_{j=2}^{i} \mathcal{J}_{j-1} T_{J_j} = \mathcal{T}_i \mathcal{J}_{i-1} J_{i-1} T_{J_i}$$

System geometry is currently uses a custom data structure. Compatibility with commonly used URDF, SDF, and VSK file formats is on the works.

Kinematic transformations map joint space variables into operational space

$$x = k(q)$$
Differential Kinematics

Jacobians (natural orthogonal complement) matrices which map joint space velocities into operational space are also provided.

Operational space velocities

\[ t_i = J_{0i} \dot{q}_0 + J_{mi} \dot{q}_m \]

Base-spacecraft Jacobian \quad Manipulator Jacobian
General Equations of motion

Generalized Convective Inertia Matrix (Coriolis and Centrifugal)

\[ H(q) \ddot{q} + C(q, \dot{q}) \dot{q} = \tau \]

- Recursive $O(n)$ formulations to obtain the $H$ and $C$ matrices are provided.
- The user has full access to the kinematic and dynamic magnitudes.
Floating-Base dynamics

- When the base is left to react to the manipulator motion the base-vehicle motion can be discounted from the dynamics model:

\[ H^* \ddot{q}_m + c^* = \tau_m \]

Floating Inertia Matrix  \hspace{2cm} Velocity dependent terms

\[ t_{x_i} = J^*_{x_i} \dot{q}_m \]

- Floating inertias and Jacobians are also provided and a recursive \( O(n) \) formulations to solve inverse dynamic problem, for a floating base, have also been implemented.
Desired Reaction Maneuver

These functions toolkit allows us to create plants (solving the dynamic problems) and controllers.

\[
\dot{q}_m = \left[ \begin{array}{c} -H_0^{-1}H_0m \\ J_{x_i}^* \end{array} \right]^{-1} \left( \begin{array}{c} \dot{q}_0 \\ t_{x_i} \end{array} \right) - \left( \begin{array}{c} H_0^{-1} \\ J_{0x_i}H_0^{-1} \end{array} \right) \mathcal{M}' \]

Initial angular momentum

The DRM maneuver uses kinematic redundancy to imposed a desired base-spacecraft reaction.
Desired Reaction Maneuver - Video
Examples

Fixed-base workspace

Straight Path floating workspace

- Joint Limits $m_0 = 100$ kg
- Dynamic Singularity $m_0 = 100$ kg
- Joint Limits $m_0 = 500$ kg
- Dynamic Singularity $m_0 = 500$ kg
Source code and documentation available:

https://github.com/NPS-SRL/SPART

Release under a GNU GPL v3 license.
Thank you for your attention.

Questions?

SPART: an open-source simulator for spacecraft robotic arm dynamic modeling and control

Dr. Josep Virgili Llop, NPS-Mechanical and Aerospace Engineering
CPT Jerry V. Drew II, NPS-Mechanical and Aerospace Engineering
Prof. Marcello Romano, NPS-Mechanical and Aerospace Engineering