The Spacedyn
a MATLAB Toolbox for Space and Mobile Robots

Programmed by:
Motoaki Shimizu, Tadashi Hiraoka, Koichi Fujishima,
Akihide Kurosu, Kenichi Hashizume and Kazuya Yoshida

Documented by:
Kazuya Yoshida

Distributed from:
`http://www.astro.mech.tohoku.ac.jp/spacedyn/`

The Space Robotics Lab.
Department of Aeronautics and Space Engineering
Tohoku University, Japan.

Contact:
`yoshida@astro.mech.tohoku.ac.jp`

This is Version 1, Release 1

released on October 6, 1999, by Kazuya Yoshida.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The Spacedyn is a MATLAB Toolbox for the kinematic and dynamic analysis and simulation of articulated multi-body systems with a moving base. Examples of such systems are a satellite with mechanical appendages, a free-flying space robot, a wheeled mobile robot, and a walking robot, all of which makes motions in the environment with or without gravity.

This toolbox can handle open chain systems with topological tree configuration. A parallel manipulator, for example, then cannot be supported directly. A walking robot contacting on the ground with more than two legs or limbs at a time seems to form a closed chain including the ground, however, we can handle such a system with a proper model of ground contact at each contact point. Parallel manipulators can be treated with virtual cut of a kinematic chain and a corresponding virtual force model.

Some academic papers regarding this toolbox is published by Kazuya Yoshida and his co-author(s) [?]. For the technical points of this software, please consult these papers as well as the following chapters of this document.

We hope that you could find this toolbox useful.

1.1 Release Note

This is “The Spacedyn: a MATLAB toolbox for space and mobile robots” version 1.0, release 1.0, as of October 7, 1999.

This is a freeware to follow the same copyright policy of other softwares in the category termed “freeware.”

We develop and distribute this software from a purly academic motivation. You can download and use it for your academic purpose only. Any of commercial use is not allowed. You can also modify and re-distribute your version at your own responsibility, but in such a case it is an etiquette to refer this toolbox as the original work.

We do not take any responsibility on any damage around you due to this software, but we would appreciate it if you could report us any bugs or mistakes, or fallacy. Any feedback is helpful for us to improve our knowledge, and update this toolbox.

When you make an academic work by using this toolbox and publish the results in papers and technical reports, we would most appreciate it if you could report it and cite this toolbox in a proper manner.

1.2 Technical Memorandums

- We developed this toolbox motivated and inspired by **Robotics toolbox** developed by Peter I. Coke, which is available from `http://www.mathworks.com/ftp/miscv4.shtml`. We took one m-file (cross.m) and use it as the original is, but our toolbox as a whole, does not have compatibility with the Peter Coke's toolbox unfortunately.

- This toolbox is for the use with MATLAB 5.0 or higher. We use three dimensional array which is not supported in version 4 or lower.

- For mathematical symbols, we give the name of variables with more than two letters. For example, vector $\boldsymbol{r}$ and $\boldsymbol{c}$ are coded by `RR` and `cc`, respectively. This is to avoid the confusion with control variables such as i, j, k, or l, m, n, etc, which are frequently used as iteration or array counters. But there is an exception: the symbol `q` is used for the joint variable vector $\boldsymbol{q}$.

- Since MATLAB doesn't allow 0 for array index, we use `R0` and `c0` instead of `RR[0]` and `cc[0]`, for example.

- We use both the input variables and the global variables to pass the values to m-file functions. The input variables inside the braces are the variables changing time to time, such as joint angles, positions, orientations, and so on. The global variables are the ones holding constant once the model is given, such as topological description matrices, kinematic and dynamic parameters.

- We assume the system composed of $n + 1$ bodies and connected by n joints. Let the body 0 be a *reference body.* Multiple branches can attach on any single *body,* as far as the system keeps a topological tree configuration. There must be a single *joint* between two bodies. We call a terminal point or the point of interest such as manipulator hand as *endpoint.* Each body, except body 0, can have one endpoint at maximum. In this document, the terms *body* and *link* are the same.

- The toolbox allows force/torque input on (1) the centroid of the reference body, (2) each endpoint, and (3) each joint. The toolbox computes the position, velocity and acceleration of (1) the centroid of the reference body, (2) the centroid of each body, (3) each endpoint, and (4) each joint.

- Computation of input force/torque are open to user programming. You can arbitrary decide each joint as either active or passive one. If you give always zero torque, such as $\tau_i = 0$, the corresponding joint behaves as a free joint. Or if you give such a torque as:
$$\tau_i = -Kq_i - D\dot{q}_i$$
the joint behaves as a passive visco-elastic joint. You can treat even a flexible link, by modeling it as a discrete successive chain of rigid links connected by elastic joints.

  Of course, you can give any arbitrary control torque determined by your own control law, on all or arbitrary selected joints.

- We know that the Denavit-Hartenberg notation is commonly used in the field of manipulator kinematics with the advantage of unique allocation of coordinate systems with minimum parameters. But we know that the DH sometimes locates the coordinate origin away from the location of an actual joint. From the dynamics point of view, the angular velocity and the inertia tensor should be defined around the corresponding joint axis or body centroid. We then do not use the DH notation but introduce a rule to define the coordinate systems with more flexibility. Our rule locates the origin of the coordinate system on each joint and orients the primary axes so that the inertia tensor should be simpler, but admits 3 position and 3 orientation parameters among two successive coordinate systems.

- For the representation of attitude or orientation, we use 3 by 3 direction cosine matrices, coded with a symbol `A`. For example, `A0` is the direction cosines to represent the attitude of the body 0. For the other bodies, a matrix `AA` is used. The advantage of direction cosine is (1) singularity free, (2) we can easily derive Roll-Pitch-Yaw angles, Euler angles, or quartanions, and (3) it is easy to find the mathematical relationship with angular velocity.

- On the other hand, we frequently need Roll-Pitch-Yaw representation also. For RPY angles, we use the symbol `Q`. For example, in order to express the twisting angles between two coordinate systems, we consider α (roll) around x axis, β (pitch) around y axis, then γ (yaw) around z axis. The set of these angles are coded by `Qi`.

- Weak points: The SpaceDyn is not good at dealing with kinematic constraints other than joint axes. It is also weak at dealing with the problems in which a contact point is dynamically changing. For those problems, a good user programming is required to model the constraint forces.

Chapter 2

Variables Used in this Toolbox

We define the variables as listed in the following pages. In the lists, the mathematical definitions and the expression in programming codes are compared. In the programming codes, `i, j, k` are array counters, where `i, j` are 1 to an arbitrary number (usually upto n but depends on each variable, please consult chapter 3 for details), but `k` is used for `1:3` only.

Table 2.1: List of variables 1

Mathematical definition	Programming code
connection index $\boldsymbol{B}_i$	`BB( i )`
incidence matrix of directed graph $\boldsymbol{S}_{ij}$	`SS( i, j )`
incidence matrix of directed graph for body 0 $\boldsymbol{S}_{0i}$	`S0( i )`
incidence matrix of directed graph for end-links $\boldsymbol{S}_{ei}$	`SE( i )`
joint type: rotational or prismatic R or P	`J_type( i )`
link vector: from centroid of link i to joint j $\boldsymbol{c}_{ij}$	`cc( k, i, j )`
link vector: from joint i to joint j $\boldsymbol{\ell}_{ij}$	`ll( k, i, j )`
link vector: from centroid of link i to end-point $\boldsymbol{c}_{ie}$	`ce( k, i )`
link vector: from joint i to end-point $\boldsymbol{\ell}_{ie}$	`le( k, i )`
link vector: from centroid of link 0 to joint i $\boldsymbol{c}_{0i}$	`c0( k, i )`

Table 2.2: List of variables 2

Mathematical definition	Programming code
linear velocity and acceleration of the centroid of link i $\boldsymbol{v}_i$, $\dot{\boldsymbol{v}}_i$	`vv( k, i ) , vd( k, i )`
linear velocity and acceleration of the centroid of link 0 $\boldsymbol{v}_0$, $\dot{\boldsymbol{v}}_0$	`v0( k ) , vd0( k )`
angular velocity and acceleration of link i around its centroid $\boldsymbol{\omega}_i$, $\dot{\boldsymbol{\omega}}_i$	`ww( k, i ) , wd( k, i )`
angular velocity and acceleration of link 0 around its centroid $\boldsymbol{\omega}_0$, $\dot{\boldsymbol{\omega}}_0$	`w0( k ) , wd0( k )`
joint angle (or prismatic displacement) q_i	`q( i )`
velocity and acceleration of joint $\dot{q}_i$, $\ddot{q}_i$	`qd( i ) , qdd( i )`
position and orientation of end-point $\boldsymbol{p}_e$, $[\boldsymbol{a}_e\boldsymbol{s}_e\boldsymbol{n}_e]$	`Pe( k, h ) , Qe( k, h )`
(inertial) force and torque applying on the centroid of link i $\boldsymbol{F}_i$, $\boldsymbol{N}_i$	`FF( k, i ) , TT( k, i )`
force and torque applying on joint i $\boldsymbol{f}_i$, $\boldsymbol{n}_i$	`Fj( k, i ) , Tj( k, i )`
(external) force and torque applying on end-point i $\boldsymbol{f}_{ei}$, $\boldsymbol{n}_{ei}$	`Fe( k, i ) , Te( k, i )`
force and torque applying on the centroid of link 0 $\boldsymbol{F}_0$, $\boldsymbol{N}_0$	`F0( k ) , T0( k )`
joint torque $\boldsymbol{\tau}_{mi}$	`tau( i )`
total mass w	`mass`
mass of link 0 and link i m_i	`m0, m( i )`

inertia tensor of link 0 and link i
$\boldsymbol{I}_0$
$\boldsymbol{I}_i$
	`inertia0`
	`inertia( k, (i*3-2):(i*3) )`

a collection of inertia tensors $\boldsymbol{I}_i$ $[\boldsymbol{I}_1, \boldsymbol{I}_2, ..., \boldsymbol{I}_n] \in R^{3n\times3}$	`inertia( k, n*3 )`
(augmented) system inertia matrices $\boldsymbol{H}$	`HH`
(augmented) system generalized force $[\boldsymbol{F}_0{}^T, \boldsymbol{N}_0{}^T, \boldsymbol{\tau}^T]^T$	`Force`
velocity dependent non-linear force $\boldsymbol{d}$	`Force0`

Table 2.3: List of variables 3

Mathematical definition	Programming code
jacobian matrix for linear velocity of link centroid i $\boldsymbol{J}_{Ti}$	`JJ_t( k, (i*n-(n-1)):(i*n) )`
jacobian matrix for angular velocity of link centroid i $\boldsymbol{J}_{Ri}$	`JJ_r( k, (i*n-(n-1)):(i*n) )`
jacobian matrix for linear velocity of endpint i $\boldsymbol{J}_{Tei}$	`JJ_te( k, (i*n-(n-1)):(i*n) )`
jacobian matrix for angular velocity of endpoint i $\boldsymbol{J}_{Rei}$	`JJ_re( k, (i*n-(n-1)):(i*n) )`
submatrices comprising the inertia matrix $\boldsymbol{H}$ $\boldsymbol{J}_{Tg}$, $\boldsymbol{H}_{\omega}$, $\boldsymbol{H}_{\omega q}$ $\boldsymbol{H}_{q}$	`JJ_tg , HH_w , HH_wq , HH_q`
position vector of the centroid of link i $\boldsymbol{r}_i$	`RR( i )`
position vector of the centroid of link i $\boldsymbol{r}_0$	`R0`
direction cosines to represent orientation of link 0 $\boldsymbol{C}_0^T$ or ${}^I\boldsymbol{A}_0$	`A0`
coordinate transformation matrix: $\{\Sigma_i\} \rightarrow \{\Sigma_I\}$ (note: the coordinate transformation matrix is mathematically equivalent to direction cosine.) ${}^I\boldsymbol{A}_i$	`AA( k, (i*3-2):(i*3) )`
a collection of coordinate transformation matrices $[{}^I\boldsymbol{A}_1, {}^I\boldsymbol{A}_2, ..., {}^I\boldsymbol{A}_n] \in R^{3n\times 3}$	`AA( k, n*3 )`
RPY angles to represent the twists from $\{\Sigma_{B(i)}\}$ to $\{\Sigma_i\}$ $(\alpha_i, \beta_i, \gamma_i)$	`Qi( k, i )`
vector for gravitational acceleration $\boldsymbol{g}$	`Gravity`
time t	`time`

Chapter 3

Notes on the Modeling of Multibody Dynamics

3.1 Mathematical Graph Representation

In order to mathematically describe the interconnection of the bodies, we adopt a method from mathematical graph theory [1]. We simplify it with additional rules on the assignment of link and joint indices, so that we can easily and uniquely construct two types of matrices (vectors); a connection index $\boldsymbol{B}$ and incidence matrices $\boldsymbol{S}$, $\boldsymbol{S}_0$, and $\boldsymbol{S}_e$.

The procedure from indices assignment to matrix construction is summarized as follows:

1. Assign the indices of links and joints in the following manner.

   - Reference body is denoted by *link* 0.
   - The index of a *link* i in the physical connection between *link* 0 and *link* j must be $0 < i < j$.
   - One link can have multiple connections with other links. As a result of the above statement, a link i ($i > 0$) has one *lower connection* (which index is smaller than i) and zero or one or multiple *upper connection(s)* (which index is greater than i).
   - There is one single *joint* interconnecting two links.
   - Indices of joints begin from 1, and *joint* i is physically attached on *link* i.
   - Then the joint interconnecting link 0 and link 1 is *joint* 1, and the joint interconnecting link 1 and link 5 is *joint* 5, for example.

2. Connection index vector $\boldsymbol{B}$ is used to find the lower connection of a link, which exists uniquely for a link i ($i > 0$).

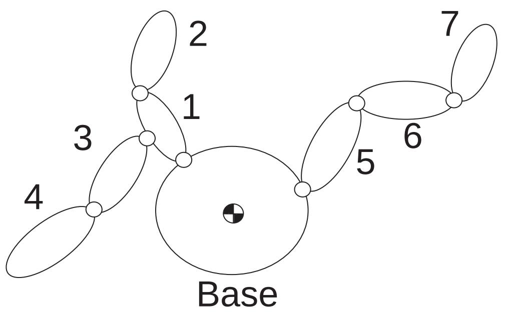


Figure 3.1: Sample System

- The element of $\boldsymbol{B}_i$ is the index of the lower connection of link i.

3. Incidence matrix $\boldsymbol{S}$ is used to find the upper connection of a link, which may not exist or exist one or more.
   Each element of $\boldsymbol{S}_{ij}$ $(i, j = 1, \dots, n)$ is defined by:
$$\boldsymbol{S}_{ij} = \begin{cases} +1 & (if \quad i = \boldsymbol{B}_j) \\ -1 & (if \quad i = j) \\ 0 & (otherwise) \end{cases}$$

4. Define a matrix $\boldsymbol{S}_{0j}$ $(j = 1, \dots, n)$ as:
$$\boldsymbol{S}_{0j} = \begin{cases} +1 & (if \quad 0 = \boldsymbol{B}_j) \\ 0 & (otherwise) \end{cases}$$
   This represents a flag to indicate if link i has a connection with the reference link 0.

5. Also define a matrix $\boldsymbol{S}_{ej}$ $(j = 1, \dots, n)$ as:
$$\boldsymbol{S}_{ej} = \begin{cases} +1 & (if \quad \text{link } j \text{ is a terminal link}) \\ 0 & (otherwise) \end{cases}$$
   This represents a flag to indicate if link i is a terminal endlink.

Figure 3.1 depicts an example of a system with multiple branches numbered by the rule presented here.

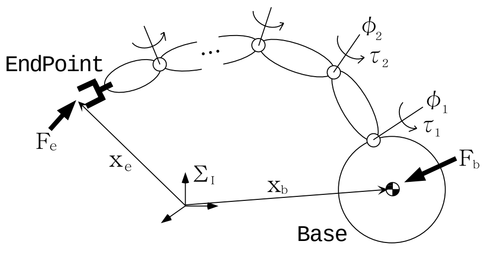


Figure 3.2: Multibody System

3.2 Coordinate System

Let the inertial reference coordinate frame be denoted by $\{\Sigma_I\}$ [1], which is stationary or lineary moving with constant velocity in the inertial space. It is not physically precise but we sometime consider the orbital fixed frame as the inertial frame in the sense of practice.

We also define moving coordinate frames fixed on each link. We do NOT take the Denavit-Hartenberg convention but introduce a simpler and flexible rule to define the link coordinates. Our rule is as follows:

1. If the joint i is revolution, then
   - locate the origin of the coordinate system $\{\Sigma_i\}$ on joint i and fixed it to the link i,
   - set its z-axis to coincide with the joint rotation axis,
   - orient its x-axis toward joint $i+1$ or the direction in which the inertia tensor is expressed easier.
2. If the joint i is prismatic, then
   - locate the origin of the coordinate system $\{\Sigma_i\}$ on the place when joint i has zero displacement and fixed it to the link $i-1$,
   - set its z-axis to coincide with the joint displacement axis, with the positive direction,
   - orient its x-axis toward the direction in which the inertia tensor is expressed easier.

[1]The expression $\{\Sigma_I\}$ is use to represent the basis of a coordinate frame, a set of unit vectors: *vectrix*, see [2]

We may also need a coordinate system located on the link centroid. In such a case, we define the link centroid coordinate i parallel to the coordinate located on joint i.

3.3 Direction Cosine and Coordinate Transformation Matrix

The direction cosine matrices $\boldsymbol{C}_i$ are commonly used to represent the attitude or orientation of body i in the inertial frame in the field of aseospace engineering [2]. On the other hand, the coordinate transformation matrices with the notation of ${}^I\boldsymbol{A}_i$ are commonly used in the field of robotics. These two are eventually the same:

$$\boldsymbol{C}_i^T = {}^I\boldsymbol{A}_i$$

Since we define the link coordinate system as above, we generally need three axis rotations to coincide from $\{\Sigma_{i-1}\}$ to $\{\Sigma_i\}$. Let $C_1(\alpha_i)$, $C_2(\beta_i)$, $C_3(\gamma_i)$ be coordinate transformation (direction cosine) around each principle axis and $C_3(q_i)$ represents the coordinate transformation by angle q_i around joint i, then we obtain the following relationship (see Figure 3.3):

$$\begin{aligned}\{\Sigma_i\} &= {}^i\boldsymbol{C}_{i-1}\{\Sigma_{i-1}\} \\ &= [C_3(q_i)C_3(\gamma_i)C_2(\beta_i)C_1(\alpha_i)]^T\{\Sigma_{i-1}\} \qquad (3.1)\end{aligned}$$

where $C_3(\gamma_i)$ and $C_3(q_i)$ seem duplicated, but γ_i corresponds to an offset angle and should be separated from a net rotation angle q_i.

Note that the RPY representation of the attitude of link 0 is:

$$\begin{aligned}\{\Sigma_0\} &= \boldsymbol{C}_0\{\Sigma_I\} \\ &= {}^0\boldsymbol{A}_I\{\Sigma_I\} \\ &= C_3(\gamma_0)C_2(\beta_0)C_1(\alpha_0)\{\Sigma_I\} \qquad (3.2)\end{aligned}$$

where $\alpha_0, \beta_0, \gamma_0$ are Roll, Pitch, Yaw angles respectively.

The direction cosines are redundant way to represent attitude, but its advantage is that the relationship between attitude and angular velocity can be expressed by a simple equation, such that:

$$\dot{\boldsymbol{C}}_0 = -\tilde{\boldsymbol{\omega}}_0\boldsymbol{C}_0 \qquad (3.3)$$

where $\dot{\boldsymbol{C}}_0$ is a time derivative of $\boldsymbol{C}_0$ and $\tilde{\boldsymbol{\omega}}$ is a skew-symmetric form of the angular velocity $\boldsymbol{\omega}_0$. This relationship is used for the routine of singularity-free integration from angular velocity to attitude.

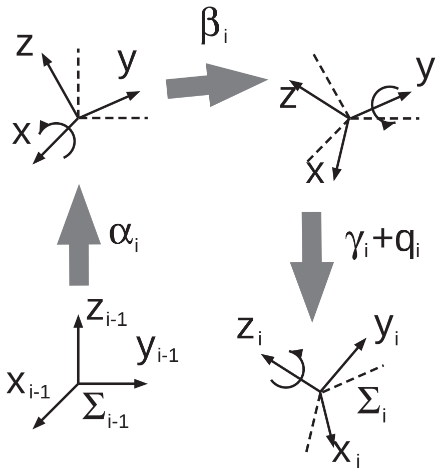


Figure 3.3: Coordinate Transformation

An additional note should be made for numerical operation in practice, the following expression which is known as the Rodorigues fomula at infinitesimal rotation provides smaller error than (3.3):

$$^{I}\boldsymbol{A}_0(t+\Delta t) = \{\boldsymbol{E} + \sin\theta_0\tilde{\boldsymbol{\omega}}_0 + (1-\cos\theta_0)\tilde{\boldsymbol{\omega}}_0^T\tilde{\boldsymbol{\omega}}_0\}^{I}\boldsymbol{A}_0(t) \tag{3.4}$$

where

$$\theta_0 = |\boldsymbol{\omega}_0\Delta t|. \tag{3.5}$$

3.4 Kinematics

3.4.1 Link Vectors

Link vectors for a link i are defined as follows (see Figure 3.4).

$\boldsymbol{c}_{ij}$: vector from the centroid of link i to joint j.

$\boldsymbol{\ell}_{ij}$: vector from joint i to joint j.

$$\boldsymbol{\ell}_{ij} = \boldsymbol{c}_{ij} - \boldsymbol{c}_{ii} \tag{3.6}$$

$\boldsymbol{c}_{ie}$: vector from the centroid of link i to the end-point if link i is an end-link.

$\boldsymbol{\ell}_{ie}$: vector from joint i to the end-point.

$$\boldsymbol{\ell}_{ie} = \boldsymbol{c}_{ie} - \boldsymbol{c}_{ii} \tag{3.7}$$

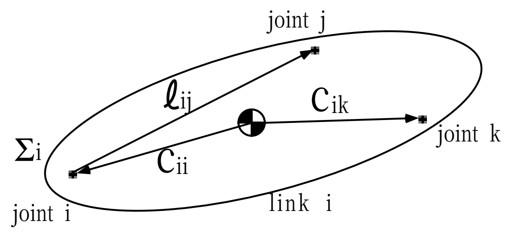


Figure 3.4: Position vectors

3.4.2 Revolution Joint

For a successive set of links connected by a revolution joint, velocity $\boldsymbol{v}_i$ and angular velocity $\boldsymbol{\omega}_i$ are calculated recursively (see Figure 3.5). When $\boldsymbol{v}_0$ and $\boldsymbol{\omega}_0$ are given,

$$
\begin{aligned}
{}^I\boldsymbol{\omega}_i &= {}^I\boldsymbol{\omega}_{B_i} + {}^I\boldsymbol{A}_i{}^i\boldsymbol{k}_i\dot{\phi}_i && (3.8)\\
{}^I\boldsymbol{v}_i &= {}^I\boldsymbol{v}_{B_i} + {}^I\boldsymbol{\omega}_{B_i} \times {}^I\boldsymbol{c}_{B_i i} - {}^I\boldsymbol{\omega}_i \times {}^I\boldsymbol{c}_{ii} && (3.9)
\end{aligned}
$$

And accelerations are calculated as the following.

$$
\begin{aligned}
{}^I\dot{\boldsymbol{\omega}}_i &= {}^I\dot{\boldsymbol{\omega}}_{B_i} + {}^I\boldsymbol{\omega}_i \times ({}^I\boldsymbol{A}_i{}^i\boldsymbol{k}_i\dot{\phi}_i) + {}^I\boldsymbol{A}_i{}^i\boldsymbol{k}_i\ddot{\phi}_i && (3.10)\\
{}^I\dot{\boldsymbol{v}}_i &= {}^I\dot{\boldsymbol{v}}_{B_i} + {}^I\dot{\boldsymbol{\omega}}_{B_i} \times {}^I\boldsymbol{c}_{B_i i} + {}^I\boldsymbol{\omega}_{B_i} \times ({}^I\boldsymbol{\omega}_{B_i i} \times {}^I\boldsymbol{c}_{B_i i})\\
&\quad -{}^I\dot{\boldsymbol{\omega}}_i \times {}^I\boldsymbol{c}_{ii} - {}^I\boldsymbol{\omega}_i \times ({}^I\boldsymbol{\omega}_i \times {}^I\boldsymbol{c}_{ii}) && (3.11)
\end{aligned}
$$

3.4.3 Prismatic Joint

If a joint is prismatic, the kinematic relationship becomes as follwos, for verocities:

$$
\begin{aligned}
{}^I\boldsymbol{\omega}_i &= {}^I\boldsymbol{\omega}_{B_i} && (3.12)\\
{}^I\boldsymbol{v}_i &= {}^I\boldsymbol{v}_{B_i} + {}^I\boldsymbol{\omega}_{B_i} \times {}^I\boldsymbol{c}_{B_i i} - {}^I\boldsymbol{\omega}_i \times {}^I\boldsymbol{c}_{ii}\\
&\quad +{}^I\boldsymbol{\omega}_i \times ({}^I\boldsymbol{A}_i{}^i\boldsymbol{k}_i\phi_i) + {}^I\boldsymbol{A}_i{}^i\boldsymbol{k}_i\dot{\phi}_i && (3.13)
\end{aligned}
$$

And for accelerations:

$$
\begin{aligned}
{}^I\dot{\boldsymbol{\omega}}_i &= {}^I\dot{\boldsymbol{\omega}}_{B_i} && (3.14)\\
{}^I\dot{\boldsymbol{v}}_i &= {}^I\dot{\boldsymbol{v}}_{B_i} + {}^I\dot{\boldsymbol{\omega}}_{B_i} \times \boldsymbol{c}_{B_i i} + {}^I\boldsymbol{\omega}_{B_i} \times ({}^I\boldsymbol{\omega}_{B_i} \times \boldsymbol{c}_{B_i i})\\
&\quad -{}^I\dot{\boldsymbol{\omega}}_i \times \boldsymbol{c}_{ii} - {}^I\boldsymbol{\omega}_i \times ({}^I\boldsymbol{\omega}_i \times \boldsymbol{c}_{ii}) + {}^I\dot{\boldsymbol{\omega}}_i \times ({}^I\boldsymbol{A}_i{}^i\boldsymbol{k}_i\phi_i)\\
&\quad +{}^I\boldsymbol{\omega}_i \times ({}^I\boldsymbol{\omega}_i \times ({}^I\boldsymbol{A}_i{}^i\boldsymbol{k}_i\phi_i)) + 2{}^I\boldsymbol{\omega}_i \times ({}^I\boldsymbol{A}_i{}^i\boldsymbol{k}_i\phi_i)\\
&\quad +{}^I\boldsymbol{A}_i{}^i\boldsymbol{k}_i\ddot{\phi}_i && (3.15)
\end{aligned}
$$

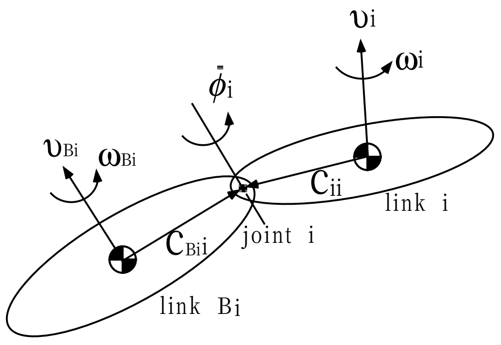


Figure 3.5: Kinematics

3.4.4 End-Point Kinematics

The kinematic relationship around the end-points is expressed as follows:

$$\dot{\boldsymbol{x}}_h = \boldsymbol{J}_m\dot{\boldsymbol{\phi}} + \boldsymbol{J}_b\dot{\boldsymbol{x}}_b \tag{3.16}$$

$$\ddot{\boldsymbol{x}}_h = \boldsymbol{J}_m\ddot{\boldsymbol{\phi}} + \dot{\boldsymbol{J}}_m\dot{\boldsymbol{\phi}} + \boldsymbol{J}_b\ddot{\boldsymbol{x}}_b + \dot{\boldsymbol{J}}_b\dot{\boldsymbol{x}}_b \tag{3.17}$$

$\boldsymbol{x}_b$	$\in R^6$	: position/orientation of the base
$\boldsymbol{x}_h$	$\in R^6$	: position/orientation of the end-points
$\boldsymbol{\phi}$	$\in R^n$	: joint variables
$\boldsymbol{J}_b$	$\in R^{6\times 6}$	: Jacobian matrix for base variables
$\boldsymbol{J}_m$	$\in R^{6\times n}$	: Jacobian matrix for joint variables

3.5 Equation of Motion

The equation of motion of the system is expressed in the following form [3, 4]:

$$\begin{bmatrix} \boldsymbol{H}_b & \boldsymbol{H}_{bm} \\ \boldsymbol{H}_{bm}^T & \boldsymbol{H}_m \end{bmatrix} \begin{bmatrix} \ddot{\boldsymbol{x}}_b \\ \ddot{\boldsymbol{\phi}} \end{bmatrix} + \begin{bmatrix} \boldsymbol{c}_b \\ \boldsymbol{c}_m \end{bmatrix} = \begin{bmatrix} \mathcal{F}_b \\ \boldsymbol{\tau} \end{bmatrix} + \begin{bmatrix} \boldsymbol{J}_b^T \\ \boldsymbol{J}_m^T \end{bmatrix} \mathcal{F}_h \tag{3.18}$$

where

$$\boldsymbol{H}_b \in R^{6\times 6} \equiv \begin{bmatrix} w\boldsymbol{E} & w\tilde{\boldsymbol{r}}_{0g}^T \\ w\tilde{\boldsymbol{r}}_{0g} & \boldsymbol{H}_\omega \end{bmatrix} \tag{3.19}$$

$$\boldsymbol{H}_{bm} \in R^{6\times n} \equiv \begin{bmatrix} \boldsymbol{J}_{Tw} \\ \boldsymbol{H}_{\omega\phi} \end{bmatrix} \tag{3.20}$$

$$\boldsymbol{H}_\omega \in R^{3\times 3} \equiv \sum_{i=1}^{n}(\boldsymbol{I}_i + m_i\tilde{\boldsymbol{r}}_{0i}^T\tilde{\boldsymbol{r}}_{0i}) + \boldsymbol{I}_0 \tag{3.21}$$

$$\boldsymbol{H}_{\omega\phi} \in R^{3\times n} \equiv \sum_{i=1}^{n}(\boldsymbol{I}_i\boldsymbol{J}_{Ri} + m_i\tilde{\boldsymbol{r}}_{0i}\boldsymbol{J}_{Ti}) \tag{3.22}$$

$$\boldsymbol{H}_m \in R^{n\times n} \equiv \sum_{i=1}^{n}(\boldsymbol{J}_{Ri}^T\boldsymbol{I}_i\boldsymbol{J}_{Ri} + m_i\boldsymbol{J}_{Ti}^T\boldsymbol{J}_{Ti}) \tag{3.23}$$

$$\boldsymbol{J}_{Tw} \in R^{3\times n} \equiv \sum_{i=1}^{n} m_i\boldsymbol{J}_{Ti}/w \tag{3.24}$$

$$\begin{aligned}\boldsymbol{J}_{Ti} \in R^{3\times n} \;\;&\equiv\;\; [\boldsymbol{k}_1 \times (\boldsymbol{r}_i - \boldsymbol{p}_1), \boldsymbol{k}_2 \times (\boldsymbol{r}_i - \boldsymbol{p}_2), \ldots, \\ &\;\;\;\;\;\ldots, \boldsymbol{k}_i \times (\boldsymbol{r}_i - \boldsymbol{p}_i), \mathbf{o}, \ldots, \mathbf{o}]\end{aligned} \tag{3.25}$$

$$\boldsymbol{J}_{Ri} \in R^{3\times n} \equiv [\boldsymbol{k}_1, \boldsymbol{k}_2, \ldots, \boldsymbol{k}_i, \mathbf{o}, \ldots, \mathbf{o}] \tag{3.26}$$

$$\boldsymbol{r}_{0g} \in R^3 \equiv \boldsymbol{r}_g - \boldsymbol{r}_0 \tag{3.27}$$

$$\boldsymbol{r}_{0i} \in R^3 \equiv \boldsymbol{r}_i - \boldsymbol{r}_0 \tag{3.28}$$

m_i	: mass of link i of arm k
w	: total mass of the system ($w = \sum_{i=1}^{n} m_i$)
$\boldsymbol{r}_i$	: position vector of centroid of link i
$\boldsymbol{p}_i$	: position vector of joint i
$\boldsymbol{k}_i$	: unit vector indicating joint axis direction of link i
$\boldsymbol{r}_0$	: position vector of centroid of satellite base body
$\boldsymbol{r}_g$	: position vector of a total centroid of the system
$\boldsymbol{c}_b, \boldsymbol{c}_m$	: velocity dependent non-linear terms
$\mathcal{F}_b$	: external force/moment on the base
$\boldsymbol{\tau}$	: joint torque of the arm
$\mathcal{F}_h$	: external force/moment on the hand
$\boldsymbol{E}$	: 3×3 identity matrix

and a tilde operator stands for a cross product such that $\tilde{\boldsymbol{r}}\boldsymbol{a} \equiv \boldsymbol{r} \times \boldsymbol{a}$. All position and velocity vectors are defined with respect to the inertial reference frame.

3.6 Forward Dynamics: Simulation Procedure

The procedure to compute a forward dynamics solutions are summarized as follows:

1. At time t, compute link positions and velocities, recurcively from link 0 to n.

2. Compute the inertia matrices using equations (17)-(26).

3. Set accelerations $\ddot{\boldsymbol{x}}_b$ and $\ddot{\boldsymbol{\phi}}$ zero, and external forces $\mathcal{F}_b$ and $\mathcal{F}_h$ zero, then compute the inertial forces recursively from link n to 0. The resultant forces on the coordinates $\boldsymbol{x}_b$ and $\boldsymbol{\phi}$ are equal to the non-liner forces $\boldsymbol{c}_b$ and $\boldsymbol{c}_m$, respectively.

4. Determine joint control forces $\boldsymbol{\tau}$ and thruster forces on the base $\mathcal{F}_b$ from a control law.

5. Compute the accelerations by:

$$\begin{bmatrix} \ddot{\boldsymbol{x}}_b \\ \ddot{\boldsymbol{\phi}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{H}_b & \boldsymbol{H}_{bm} \\ \boldsymbol{H}_{bm}^T & \boldsymbol{H}_m \end{bmatrix}^{-1} \left\{ \begin{bmatrix} \mathcal{F}_b \\ \boldsymbol{\tau} \end{bmatrix} + \begin{bmatrix} \boldsymbol{J}_b^T \\ \boldsymbol{J}_m^T \end{bmatrix} \mathcal{F}_h - \begin{bmatrix} \boldsymbol{c}_b \\ \boldsymbol{c}_m \end{bmatrix} \right\} \tag{3.29}$$

6. Integrate the above accelerations to yield the velocities and positions at time $t + \Delta t$.

7. go to 1. and continue.

3.7 Inverse Dynamics

Inverse dynamic computation is useful for a computed torque control. It is also needed for the forward dynamics in numerical computation the velocity dependent non-linear terms as described in the last section.

For the inverse dynamic computation, an order-n, recursive Newton-Euler approach [5] is well-known.

Newton and Euler equations for a link i are expressed as:

$$\boldsymbol{F}_i = m_i \dot{\boldsymbol{v}}_i \tag{3.30}$$
$$\boldsymbol{N}_i = \boldsymbol{I}_i \dot{\boldsymbol{\omega}}_i + \boldsymbol{\omega}_i \times (\boldsymbol{I}_i \boldsymbol{\omega}_i) \tag{3.31}$$

where $\boldsymbol{F}_i, \boldsymbol{N}_i$ are inertial force and moment exert on the link centroid. Together with the following force and moment exerting on the joint or end-point,

$\boldsymbol{f}_i, \boldsymbol{n}_i$: Force and moment on joint i.

$\boldsymbol{f}_{ci}, \boldsymbol{n}_{ci}$: Force and momnet on end-point (if link i is an end-link) the dynamic equilibrium expressed in the following form (see Figure 3.6):

$$\begin{aligned}
\boldsymbol{f}_i &= \boldsymbol{F}_i + \sum_{j=i+1}^{n} \boldsymbol{S}_{ij}\boldsymbol{f}_j + \boldsymbol{S}_{ei}\boldsymbol{f}_{ei} && (3.32)\\
\boldsymbol{n}_i &= \boldsymbol{N}_i + \sum_{j=i+1}^{n} \boldsymbol{S}_{ij}(\boldsymbol{\ell}_{ij} \times \boldsymbol{f}_j + \boldsymbol{n}_j)\\
&\quad + \boldsymbol{S}_{ii}\boldsymbol{c}_{ii} \times \boldsymbol{F}_i + \boldsymbol{S}_{ei}(\boldsymbol{\ell}_{ie} \times \boldsymbol{f}_{ei} + \boldsymbol{n}_{ei}) && (3.33)
\end{aligned}$$

for around a revolution joint, and

$$\begin{aligned}
\boldsymbol{f}_i &= \boldsymbol{F}_i + \sum_{j=i+1}^{n} \boldsymbol{S}_{ij}\boldsymbol{f}_j + \boldsymbol{S}_{ei}\boldsymbol{f}_{ei} && (3.34)\\
\boldsymbol{n}_i &= \boldsymbol{N}_i + \sum_{j=i+1}^{n} \boldsymbol{S}_{ij}(\boldsymbol{\ell}_{ij} \times \boldsymbol{f}_j + \boldsymbol{n}_j) && (3.35)\\
&\quad + \boldsymbol{S}_{ii}(\boldsymbol{c}_{ii} - \phi_i) \times \boldsymbol{F}_i + \boldsymbol{S}_{ei}(\boldsymbol{\ell}_{ie} \times \boldsymbol{f}_{ei} + \boldsymbol{n}_{ei})
\end{aligned}$$

for around a prismatic joint.

After the computation of whole $\boldsymbol{f}_i$ and $\boldsymbol{n}_i$ for $i = 1$ to n, we can obtain joint torque as:

$$\begin{aligned}
\boldsymbol{\tau}_i &= \boldsymbol{n}_i^{TI}\boldsymbol{k}_i \quad (if \quad \text{revolution joint}) && (3.36)\\
\boldsymbol{\tau}_i &= \boldsymbol{f}_i^{TI}\boldsymbol{k}_i \quad (if \quad \text{prismatic joint}) && (3.37)
\end{aligned}$$

And the reaction force/moment on the base centroid is obtained as follows:

$$\begin{aligned}
\boldsymbol{F}_0 &= \sum_{i=1}^{n} \boldsymbol{S}_{0i}\boldsymbol{f}_i && (3.38)\\
\boldsymbol{N}_0 &= \sum_{i=1}^{n} \boldsymbol{S}_{0i}(\boldsymbol{c}_{0i} \times \boldsymbol{f}_i + \boldsymbol{n}_i) && (3.39)
\end{aligned}$$

3.8 Application Examples

Here, some of applications for dynamic simulation of moving-base sytems are illustrated, which all are relevant to actual space flight missions.

Figure 7 (a) depicts a simulation model of ETS-VII, a Japanese free-flying space robot with 2 meter long 6 DOF manipulator arm. The satellite was launched November, 1997. It is currently flying in orbit, as of August 1998, and a number of significant experiments on space robotics are conducting on the satellite. Free-flying system dynamics including

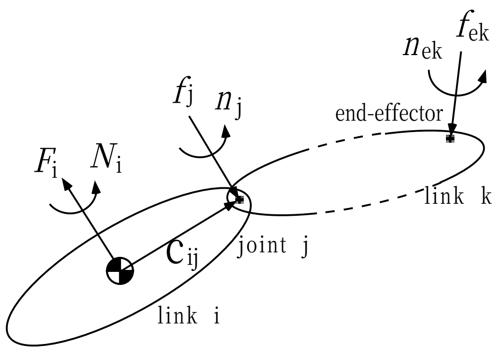


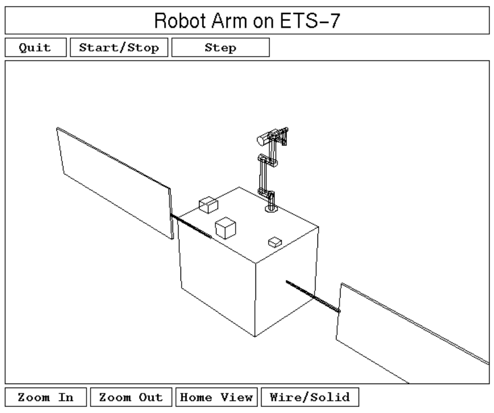
Figure 3.6: Dynamic equilibrium

manipulator reaction and the vibrations of solar paddles can be analyzed with the SpaceDyn toolbox.

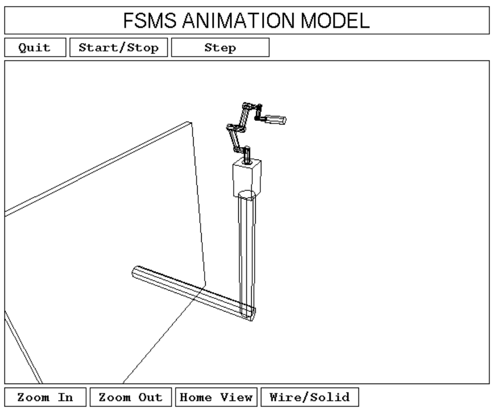
Figure 7 (b) depicts a flexible-base robot. Practical examples of such a system are SRMS-SPDM system, a Canadian made space station manipulator system and JEMRMS, a macro-mini manipulator system for the Japanese Experimental Module of the station. For these systems the internal dynamics, as presented in the following section, will be a key technology in terms of the reaction and vibration management.

Both figures 7 (a) and (b) are illustrated using a useful animation tool named "XAnimate, " which can be freely downloaded from [6].

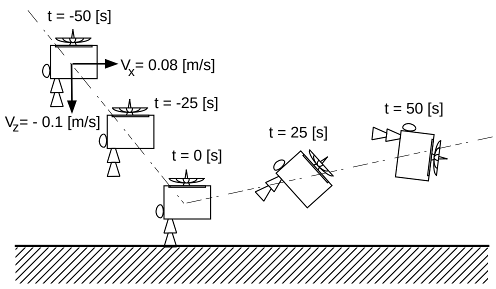
Figure 7 (c) is a touch-down simulation of MUSES-C asteroid sample-return spacecraft. For this simulation, impulsive ground contact is a key issue and the contact model discussed in the previous subsection is applied [7]. With the development of the contact model for tire mechanics, the dynamic motion of off-road articulated vehicle can be also simulated, as shown in (d) [8], such an application is found in a mission of a planetary exploration rover.



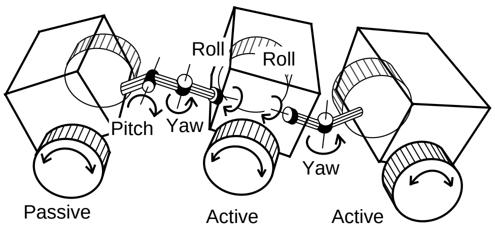
(a) A simulation model for Free-Flying Space Robot



(b) A simulation model for Flexible-Base Robot



(c) Touch-down simulation of MUSES-C Asteroid Sample-Return Spacecraft



(d) An example of an articulated off-road vehicle as a potential design of a planetary rover

Figure 7 Practical applications of the dynamics simulation of moving-base robots by "SpaceDyn" toolbox

Bibliography

[1] Jens Wittenburg: *Dynamics of Systems of Rigid Bodies,* B. G. Teubner Stuttgart, 1977.

[2] Peter C. Hughes: *Spacecraft Attitude Dynamics,* John Wiley & Sons, Inc., 1986.

[3] *Space Robotics: Dynamics and Control,* edited by Xu and Kanade, Kluwer Academic Publishers, 1993.

[4] K. Yoshida: "A General Formulation for Under-Actuated Manipulators," *Proc. 1997 IEEE/RSJ Int. Conf. on Intelligent Robots and Syatems,* pp.1651-1957, Grenoble, France, 1997.

[5] J. S. Y. Luh, M. W. Walker and R. P. C. Paul: "On-Line Computational Scheme for Mechanical Manipulators," *Trans. ASME J. Dynamic Systems, Measurements and Control,* vol.120, pp.69-76, 1980.

[6] ftp://eeftp.eng.ohio-state.edu/pub/orin/

[7] K. Yoshida and T. Hiraoka, "Touch Down Simulation of the MUSES-C Satellite for Asteroid Sampling," AIAA Modeling & Simulation Technologies Conf. Paper 98–4376, Boston, MA, August, 1998.

[8] K. Yoshida, T. Shiwa, and M. Oda, "Dynamic Simulation of an Articulated Off-Road Vehicle," AIAA Modeling & Simulation Technologies Conf. Paper 98–4362, Boston, MA, August, 1998.