Copy No. _____

Defence Research and Development Canada
Recherche et développement pour la défense Canada

DEFENCE R&D DÉFENSE

Ship Mechanical Systems API – Final Report: Modeling Methodologies, API User Manual, API Validation

A. Roy
D. Steinke
R. Nicoll
Dynamic Systems Analysis Limited

Prepared by:
Dynamic Systems Analysis Limited
101-19 Dallas Road, Victoria, BC, Canada V8V 5A6

Project Manager: Dean Steinke, 902-407-3722
Contract Number: W7707-115188/001/HAL
Contract Scientific Authority: Kevin McTaggart, 902-426-3100 ext 253

Defence R&D Canada – Atlantic

Contract Report
DRDC Atlantic CR 2010-256
January 2014

Canada

This page intentionally left blank.

Ship Mechanical Systems API – Final Report: Modeling Methodologies, API User Manual, API Validation

A. Roy
D. Steinke
R. Nicoll

Prepared by:

Dynamic Systems Analysis Limited
101-19 Dallas Road, Victoria, BC, Canada, V8V 5A6

Project Manager: Dean Steinke 902-407-3722
Contract Number: W7707-115188/001/HAL
Contract Scientific Authority: Kevin McTaggart 902-426-3100 ext. 253

Defence Research and Development Canada – Atlantic
Contract Report
DRDC Atlantic CR 2010-256
January 2014

Abstract

This report describes development of a new software library for modelling ship mechanical systems. The library can model cables, winches, cables, and payloads. The library is also able to model collections of these components working together. When developing the library, computational fidelity was of primary importance and computational speed was of secondary importance. This initial version of the software library includes modelling of collisions and associated contact forces. The library is intended for modelling of naval platform systems, such as launch and recovery of a small craft from a ship.

Résumé

Le présent rapport décrit l'élaboration d'une nouvelle bibliothèque de logiciels pour la modélisation de systèmes mécaniques de navires. Celle-ci permet de modéliser câbles, treuils et charges utiles, ainsi que des regroupements de ces composants en interaction. Au cours du développement de cette bibliothèque, la fiabilité de traitement constituait un élément de première importance, alors que la vitesse de calcul était secondaire. Cette version initiale comprend une fonctionnalité de modélisation des collisions ainsi que des forces de contact connexes. Elle a été conçue en vue de modéliser des systèmes de plateforme navale, comme le lancement et la récupération d'un petit bâtiment à partir d'un navire.

This page intentionally left blank.

Executive summary

Ship Mechanical Systems API – Final Report: Modeling Methodologies, API User Manual, API Validation

A. Roy, D. Steinke, R. Nicoll; DRDC Atlantic CR 2010-256; Defence Research and Development Canada – Atlantic; January 2014.

Background: Advances in computational hardware and software are enabling simulation of complex systems. Such simulations can be used for naval platforms applications, such as analysis of ship launch and recovery.

Principal results: This report describes development of a new software library for modelling ship mechanical systems. The library can model cables, winches, cables, and payloads. The library is also able to model collections of these components working together. When developing the library, computational fidelity was of primary importance and computational speed was of secondary importance. This initial version of the software library includes modelling of collisions and associated contact forces.

Significance of results: The developed software library can form the foundation for simulation of naval platform systems, including replenishment and launch and recovery. These simulations can be used to reduce economic and operational risks.

Future work: The ship mechanical systems software library used to develop simulations of systems, including a ship towing another vessel and launch and recovery of a small craft from a ship.

Sommaire

Ship Mechanical Systems API – Final Report: Modeling Methodologies, API User Manual, API Validation

A. Roy, D. Steinke, R. Nicoll ; DRDC Atlantic CR 2010-256 ; Recherche et développement pour la défense Canada – Atlantique ; janvier 2014.

Introduction : Les avancées en matière de logiciels et de matériel de calcul permettent d'effectuer des simulations de systèmes complexes. Celles-ci peuvent servir dans le cadre d'applications de plateforme navale, par exemple pour l'analyse du lancement et de la récupération de navire.

Principaux résultats : Le présent rapport décrit l'élaboration d'une nouvelle bibliothèque de logiciels pour la modélisation de systèmes mécaniques de navires. Celle-ci permet de modéliser câbles, treuils et charges utiles, ainsi que des regroupements de ces composants en interaction. Au cours du développement de cette bibliothèque, la fiabilité de traitement constituait un élément de première importance, alors que la vitesse de calcul était secondaire. Cette version initiale comprend une fonctionnalité de modélisation des collisions ainsi que des forces de contact connexes.

Importance des résultats : Cette bibliothèque de logiciels peut servir de fondement à la simulation de systèmes de plateforme navale, y compris le ravitaillement ainsi que le lancement et la récupération. On peut ensuite employer celle-ci afin d'atténuer les risques économiques et opérationnels.

Plans futurs : Employer la bibliothèque de logiciels de systèmes mécaniques de navire pour développer des simulations de systèmes, notamment un navire qui en remorque un autre ou encore le lancement et la récupération d'un petit bâtiment à partir d'un navire.

Table of contents

List of figures

List of tables

List of Symbols

General Symbols

E	The Earth-fixed frame, the fixed inertial frame
$\hat{\mathbf{X}}$	The $\hat{\mathbf{X}}$ axis of the Earth-fixed frame
$\hat{\mathbf{Y}}$	The $\hat{\mathbf{Y}}$ axis of the Earth-fixed frame
$\hat{\mathbf{Z}}$	The $\hat{\mathbf{Z}}$ axis of the Earth-fixed frame
$\hat{\mathbf{x}}$	The $\hat{\mathbf{x}}$ axis of any Cartesian frame besides the Earth-fixed frame
$\hat{\mathbf{y}}$	The $\hat{\mathbf{y}}$ axis of any Cartesian frame besides the Earth-fixed frame
$\hat{\mathbf{z}}$	The $\hat{\mathbf{z}}$ axis of any Cartesian frame besides the Earth-fixed frame
$O(N)$	Big O notation, used to define the limiting behaviour of a function as the size of the argument N grows.
dt	The differential time, generally assumed to be the integration step size.

Cable Theory Symbols

$\mathbf{r}(s)$	The position in Earth-fixed frame coordinates of a location s along the cable.
s	The unstretched arc length, the distance to some point on the cable measured along the cable starting at node 0.
$s^{(i)}$	The unstretched arc length of node i.
$\mathbf{r}^{(i)}$	The Earth-fixed frame coordinates of node i.
$\mathbf{r}''$	The second spatial derivative of Earth-fixed frame coordinates of node i, akin to $\frac{d^2\mathbf{r}}{ds^2}$.
$\mathbf{r}(s,t)$	The position in Earth-fixed frame coordinates of a location s along the cable at some instance in time t.

$\phi_{i,j}$	The shape function j for node i.
$\hat{\mathbf{t}}$	The axis of the Frenet frame which represents the cable tangent, akin to $\frac{d\mathbf{r}}{ds}$.
$\hat{\mathbf{n}}$	The axis of the Frenet frame which represents the cable normal, points toward the center of curvature of the cable.
$\hat{\mathbf{b}}$	The axis of the Frenet frame which represents the cable binormal.
$\hat{\mathbf{P}}_{10}$	The normal axis of the mechanical frame rotated from the Frenet frame by an amount α about the tangent axis.
$\hat{\mathbf{P}}_{20}$	The binormal axis of the mechanical frame rotated from the Frenet frame by an amount α about the tangent axis.
$\hat{\mathbf{q}}$	The tangent axis of the mechanical frame rotated from the Frenet frame by an amount α about the tangent axis. Co-linear with $\hat{\mathbf{t}}$
α	The twist angle of the cable mechanical frame relative to the Frenet frame.
$Lu^{(i)}$	The undeformed length of cable element i.
h	The hydrodynamic force distribution along a cable element.
w	The gravitational force distribution along a cable element.
ρ	The radius of curvature
$\mathbf{C}$	The center of curvature.
$\mathbf{F}^{(i)}$	The internal linear force in the cable at node i
$\mathbf{M}^{(i)}$	The internal moment in the cable at node i
$\mathbf{K}_{\epsilon}$	The cable axial stiffness.
$\mathbf{K}_{\kappa}$	The cable bending stiffness.
$\mathbf{K}_{\tau}$	The cable torsional stiffness.

$\mathbf{K}_{\epsilon G}$	The cable axial stiffness with the lumped mass approximation applied.
$\mathbf{K}_{\kappa G}$	The cable bending stiffness with the lumped mass approximation applied.
$\mathbf{K}_{\tau G}$	The cable torsional stiffness with the lumped mass approximation applied.
X	The assembly of node positions $\mathbf{r}$ and curvatures $\mathbf{r}''$.
$\ddot{X}$	The assembly of second temporal derivatives of node positions $\mathbf{r}$ and curvatures $\mathbf{r}''$.
W	The weight and buoyancy load.
H	The hydrodynamic load.
W_G	The weight and buoyancy load with the lumped mass approximation applied.
H_G	The hydrodynamic load with the lumped mass approximation applied.
M_c	Consistent element mass matrix.
M_G	Mass matrix with row elements relevant to curvature removed.
$\mathbf{R}_G$	The cable state consisting only of node positions.
$\ddot{\mathbf{R}}_G$	The second time derivative of the cable state consisting only of node positions.
τ	Mechanical frame torsion of cable.
γ	Frenet frame torsion of cable.
γ_G	Vector of Frenet frame torsion of cable.
GJ	Torsional stiffness cable property.
EI	Bending stiffness cable property.

EA	Axial stiffness cable property.
$\mathbf{L}_G$	Tridiagonal matrix based on element lengths.
α_G	A vector of torsional deformations from torsional frame to mechanical frame.
$\mathbf{T}^{(i)}$	External torque applied to node i

Winch Theory Symbols

$v^w_{(t_k)}$	The winch payout rate at time t_k
a^w_{accel}	The winch payout acceleration setpoint.
a^w_{deccel}	The winch payout deceleration setpoint.
v^w_{set}	The target velocity setpoint.

Payload Theory Symbols

${}^E\mathbf{F}$	A 6×1 spatial vector containing the force and moment applied to the rigid body, expressed with respect to the Earth-fixed frame.
${}^E\mathbf{M}$	The 6×6 mass-inertia matrix expressed with respect to the Earth-fixed frame.
${}^E\mathbf{a}_{CG}$	The 6×1 spatial vector containing the linear and angular acceleration of the rigid body frame CG expressed with respect to the Earth-fixed frame.
m	The mass of the rigid body.
I_{xx}	The mass moment of inertia about the $\hat{\mathbf{x}}$ axis.
I_{yy}	The mass moment of inertia about the $\hat{\mathbf{y}}$ axis.
I_{zz}	The mass moment of inertia about the $\hat{\mathbf{z}}$ axis.

I_{xy}	The xy product of inertia, equal to the yx product of inertia.
I_{xz}	The xz product of inertia, equal to the zx product of inertia.
I_{yz}	The yz product of inertia, equal to the zy product of inertia.
${}^{E}\mathbf{v}_{CG}$	The 6×1 spatial vector containing the linear and angular velocity of the rigid body frame CG expressed with respect to the Earth-fixed frame.
${}^{E}\check{\mathbf{P}}_{CG}$	A 6×1 array containing the 3 Cartesian position components of the rigid body frame CG with respect to the earth fixed frame and the 3 Euler angles that describe the relative orientation of the frames.
ψ	Euler angle of rotation, describes rotation about the $\hat{\mathbf{z}}$ axis.
θ	Euler angle of rotation, describes rotation about the $\hat{\mathbf{y}}'$ axis.
ϕ	Euler angle of rotation, describes rotation about the $\hat{\mathbf{x}}''$ axis.
${}^{E}\dot{\check{\mathbf{P}}}_{(CG)}(\psi)$	The 6×1 array containing the time derivative of the 3 Cartesian position components of the rigid body frame CG with respect to the earth fixed frame and the 3 Euler angles that describe the relative orientation of the frames.
${}^{E}\mathbf{T}_{B}$	The 6×6 transformation matrix that rotates a vector's frame of reference from the Payload's body-fixed frame to the Earth-fixed frame.
${}^{B}\mathbf{v}_{CG}$	The 6×1 spatial vector containing the linear and angular velocity of the rigid body frame CG expressed with respect to the body-fixed frame.
$c\phi$	$cos(\phi)$
$s\phi$	$sin(\phi)$
$c\theta$	$cos(\theta)$
$s\theta$	$sin(\theta)$
$t\theta$	$tan(\theta)$
$c\psi$	$cos(\psi)$

$s\psi$	$sin(\psi)$
$\mathbf{R}_1$	A 3×3 rotation matrix describing the body-fixed frame with respect to the Earth-fixed frame.
$\mathbf{R}_2$	A 3×3 non-orthogonal matrix, converts the angular velocity of the rigid body to the Euler angles rates of change.
${}^{B}\mathbf{a}_{cg}$	The 6×1 absolute acceleration spatial vector of the rigid body, expressed with respect to the body-fixed frame.
${}^{B}\dot{\mathbf{v}}_{cg}$	The time derivative of the 6×1 absolute velocity spatial vector of the rigid body, expressed with respect to the body-fixed frame.
${}^{B}\mathbf{\Omega}_{B}$	A 6×6 cross-product operator of ω, where $\omega = \{\omega_x, \omega_y, \omega_z\}$. When multiplied against some 6×1 vector $\mathbf{V}$, *i.e.* ${}^{B}\mathbf{\Omega}_{B}\mathbf{V}$, returns the cross-product $[\omega \times \mathbf{v}_{linear}, \omega \times \mathbf{v}_{angular}]^T$, where $\mathbf{v}_{linear}$ and $\mathbf{v}_{angular}$ are the linear and angular components of $\mathbf{V}$.
${}^{B}\mathbf{v}_{cg}$	The 6×1 absolute velocity spatial vector of the rigid body, expressed with respect to the body-fixed frame.
ω_x	The $\hat{\mathbf{x}}$ component of the angular velocity vector.
ω_y	The $\hat{\mathbf{y}}$ component of the angular velocity vector.
ω_z	The $\hat{\mathbf{z}}$ component of the angular velocity vector.
M	The 6×6 mass inertia matrix of the payload expressed with respect to the body-fixed frame.
${}^{B}\mathbf{F}$	The 6×1 external force applied to the rigid body expressed about the body-fixed frame.
$\mathbf{F}_c$	A 6×1 spatial vector representing the bias force accounting for apparent accelerations resulting from angular rotation rate of the frame.
${}^{E}\breve{\mathbf{a}}_{CG}$	A 6×1 array containing the second time derivative of the 3 Cartesian position components of the rigid body frame CG with respect to the earth fixed frame and the 3 Euler angles that describe the relative orientation of the frames.
${}^{E}\dot{\mathbf{T}}_{B}$	The time derivative of the transformation matrix ${}^{E}\mathbf{T}_{B}$.

$\dot{\mathbf{R}}_1$	The time derivative of the transformation matrix $\mathbf{R}_1$.
$\dot{\mathbf{R}}_2$	The time derivative of the transformation matrix $\mathbf{R}_2$.
$\dot{\phi}$	Euler angle rate of change.
$\dot{\theta}$	Euler angle rate of change.
$\dot{\gamma}$	Euler angle rate of change.
Ω	The 3×3 skew-symmetric cross-product operator of $\omega = \omega_x, \omega_y, \omega_z$, when multiplied to some vector, $\mathbf{v}$, returns the cross-product $\omega \times \mathbf{v}$.
${}^{B}\mathbf{P}_{A}$	The 3×1 vector describing the location of the cable attachment point with respect to the body-fixed frame.
$\hat{\mathbf{P}}_{10}$	A mechanical frame axis, rotated an angle α from the Frenet frame normal $\hat{\mathbf{n}}$ axis of the cable, at node 0, used for clamped cable connection to force the boundary connection.
$\hat{\mathbf{P}}_{20}$	A mechanical frame axis, rotated an angle α from the Frenet frame binormal $\hat{\mathbf{b}}$ axis of the cable, at node 0, used for clamped cable connection to force the boundary connection.
$\hat{\mathbf{P}}_{30}$	A mechanical frame axis, equal to the Frenet frame tangent $\hat{\mathbf{t}}$ axis of the cable, at node 0, used for clamped cable connection to force the boundary connection.
$\hat{\mathbf{P}}_{1N}$	A mechanical frame axis, equal to the Frenet frame tangent $\hat{\mathbf{n}}$ axis of the cable, at node N, used for clamped cable connection to force the boundary connection.
$\hat{\mathbf{P}}_{2N}$	A mechanical frame axis, equal to the Frenet frame tangent $\hat{\mathbf{b}}$ axis of the cable, at node N, used for clamped cable connection to force the boundary connection.
$\hat{\mathbf{P}}_{3N}$	A mechanical frame axis, equal to the Frenet frame tangent $\hat{\mathbf{t}}$ axis of the cable, at node N, used for clamped cable connection to force the boundary connection.
α_{twist}	The total amount of twist in the cable.

τ_{twist}	The mechanical torsion in the cable.
${}^{E}\mathbf{F}_{cable}$	The 6×1 spatial vector containing the force and moment the cable is applying to the payload, expressed with respect to the Earth-fixed frame.

Boomcrane Theory Symbols

θ_i	Link i's joint angle θ D-H parameter
d_i	Link i's joint length d D-H parameter
a_i	Link i's length a D-H parameter
α_i	Link i's twist α D-H parameter
$\hat{\mathbf{x}}_i$	The $\hat{\mathbf{x}}$ axis of joint frame i
$\hat{\mathbf{y}}_i$	The $\hat{\mathbf{y}}$ axis of joint frame i
$\hat{\mathbf{z}}_i$	The $\hat{\mathbf{z}}$ axis of joint frame i
$\mathbf{A}_{d_i}^{(4\times4)}$	A 4×4 transformation matrix representing the effect of joint length.
$\mathbf{A}_{\theta_i}^{(4\times4)}$	A 4×4 transformation matrix representing the effect of joint angle.
$\mathbf{A}_{\alpha_i}^{(4\times4)}$	A 4×4 transformation matrix representing the effect of link twist.
$\mathbf{A}_{a_i}^{(4\times4)}$	A 4×4 transformation matrix representing the effect of link length.
$\mathbf{A}_{i\rightarrow i-1}^{(4\times4)}$	A 4×4 transformation matrix that transforms frame i to frame $i-1$
$\mathbf{R}_{i\rightarrow i-1}$	A 3×3 rotation matrix that changes the frame of reference of a vector from frame i to $i-1$
${}^{i}\mathbf{P}_{i\rightarrow i-1}$	A 3×1 vector from frame i to frame $i-1$, expressed with respect to frame i

$\hat{\mathbf{x}}_p^{(i)}$	The $\hat{\mathbf{x}}$ axis of link i's proximal joint frame
$\hat{\mathbf{y}}_p^{(i)}$	The $\hat{\mathbf{y}}$ axis of link i's proximal joint frame
$\hat{\mathbf{z}}_p^{(i)}$	The $\hat{\mathbf{z}}$ axis of link i's proximal joint frame
p	The proximal joint frame for a link
d	The distal joint frame for a link
$\hat{\mathbf{x}}_d^{(i)}$	The $\hat{\mathbf{x}}$ axis of link i's distal joint frame
$\hat{\mathbf{y}}_d^{(i)}$	The $\hat{\mathbf{y}}$ axis of link i's distal joint frame
$\hat{\mathbf{z}}_d^{(i)}$	The $\hat{\mathbf{z}}$ axis of link i's distal joint frame
$\mathbf{R}_{(p\rightarrow E)}^{(i)}$	A 3×3 rotation matrix, belonging to link i that changes the frame of reference of a vector, from its proximal joint frame p to the Earth-fixed frame E
$\mathbf{R}_{(p\rightarrow d)}^{(i)}$	A 3×3 rotation matrix, belonging to link i that changes the frame of reference of a vector, from its proximal joint frame p to its distal joint frame frame d
$\mathbf{P}_{(E\rightarrow p)}^{(i)}$	A 3×1 vector, starting at the Earth-fixed frame, ending at link i's proximal joint frame p, assumed to be expressed with respect to link i's frame, the proximal joint frame, unless stated otherwise.
$\mathbf{P}_{(p\rightarrow d)}^{(i)}$	A 3×1 vector, starting at a link i's proximal joint frame p, ending at it's distal joint frame d, assumed to be expressed with respect to link i's frame, the proximal joint frame, unless stated otherwise.
ψ	Heading angle; rotation about the $\hat{\mathbf{z}}$ axis.
θ	Pitch angle; rotation about the $\hat{\mathbf{y}}'$ axis.
ϕ	Roll angle; rotation about the the $\hat{\mathbf{x}}''$ axis.
$\mathbf{P}_{p\rightarrow cg}^{(i)}$	A 3×1 vector, starting at a link i's proximal joint frame p, ending at its CG, assumed to be expressed with respect to link i's frame, the proximal joint frame.

$\tilde{\mathbf{P}}^{(i)}_{p\to d}$	A skew-symmetric representation of the vector $\mathbf{P}^{(i)}_{p\to d}$, used as a cross-produce operator.
$\mathbf{R}^{(i)}_{p\to CG}$	A 3×3 rotation matrix, belonging to link i that changes the frame of reference of a vector, from its proximal joint frame p to its CG frame.
$\mathbf{R}^{(i)}_{\theta}$	A 3×3 rotation matrix used to account for the change in orientation of link i's rigid body body-fixed frame relative to a revolute proximal joint's frame.
$\mathbf{R}'^{(i)}_{p\to CG}$	A 3×3 rotation matrix, describing the orientation of link i's CG frame relative to the proximal joint frame, accounting for the change in orientation caused by motion of the proximal joint.
$\mathbf{P}'^{(i)}_{p\to CG}$	A 3×1 vector, starting at link i's proximal joint frame p, ending at its CG accounting for the change in orientation or displacement caused by the motion of the proximal joint.
$\mathbf{P}^{(i)}_{displ}$	A 3×1 vector, describing the translation of the CG frame due to a prismatic proximal joint.
$\check{\mathbf{P}}^{(i)}_{p\to CG}$	A 6×1 array used to describe the position and orientation of a link's rigid body body-fixed frame CG relative to the proximal joint frame, using 3 Cartesian position components and 3 Euler angles expressed about the non-orthogonal frame $\hat{\mathbf{Z}}\hat{\mathbf{Y}}'\hat{\mathbf{X}}''$
$\phi^{(i)}_{p}$	A 6×1 array, filled with zeros except for a single entry that contains a one. It is used to map a scalar joint deflection, velocity or acceleration to a 6×1 array. The matrix $\phi^{(i)}_{p}$ is different depending on whether or not link i's proximal joint is revolute or prismatic.
$\mathbf{T}^{(v)(i)}_{p\to d}$	A 6×6 transformation matrix used to transform 6×1 spatial velocity or acceleration vectors from link i's proximal frame to its distal frame.
$\mathbf{T}^{(f)(i)}_{p\to d}$	A 6×6 transformation matrix used to transform 6×1 spatial force vectors from link i's proximal frame to its distal frame.
$\varepsilon^{(i)}_{p}$	A scalar variable describing link i's proximal joint deflection.
$\dot{\varepsilon}^{(i)}_{p}$	A scalar variable describing link i's proximal joint deflection rate.

$\ddot{\varepsilon}_p^{(i)}$	A scalar variable describing the acceleration of the deflection of link i's proximal joint.
$\mathbf{v}_p^{(i)}$	A 6×1 spatial vector representing the velocity of joint frame p of link i. It contains 3 Cartesian linear velocity components and 3 Cartesian angular velocity components of the proximal joint expressed about link i's frame.
$\mathbf{v}_d^{(i)}$	A 6×1 spatial vector representing the velocity of joint frame d of link i. It contains 3 Cartesian linear velocity components and 3 Cartesian angular velocity components of the distal joint expressed about link i's frame.
$\omega_p^{(i)}$	A 3×1 Cartesian vector describing the angular rotation rate of link i's proximal joint.
$\mathbf{a}_p^{(i)}$	A 6×1 spatial vector representing the acceleration of joint frame p of link i. It contains 3 Cartesian linear acceleration components and 3 Cartesian angular acceleration components of the proximal joint express about link i's frame.
$\mathbf{a}_d^{(i)}$	A 6×1 spatial vector representing the acceleration of joint frame d of link i. It contains 3 Cartesian linear acceleration components and 3 Cartesian angular acceleration components of the distal joint express about link i's frame.
$\mathbf{a}_{base}$	A 6×1 spatial vector representing the acceleration of a articulated body's (boom crane) base frame.
$\mathbf{M}_{CG}^{(i)}$	A 6×6 mass inertia matrix for link i's rigid body expressed about the CG frame.
$\mathbf{m}^{(i)}$	A 3×3 diagonal mass matrix for link i's rigid body expressed about the CG frame.
$\mathbf{M}_p^{(i)}$	A 6×6 mass inertia matrix for link i's rigid body expressed about the link's proximal joint frame.
$\mathbf{M}_p^{*(i)}$	The articulated inertia of the mechanism including all of the links from the last link to link i expressed about link i's proximal joint frame.
$\mathbf{I}_{CG}^{(i)}$	A 3×3 mass moment of inertia tensor for link i's rigid body about its CG frame.
$\mathbf{I}_p^{(i)}$	A 3×3 mass moment of inertia tensor for link i's rigid body about its proximal joint frame.

$m^{(i)}$	A scalar value representing the mass of a link's rigid body.
$m_p^{*(i)}$	A scalar value representing the inertial property of the articulated mass inertia matrix $\mathbf{M}_p^{*(i)}$, that acts along the proximal joint degree of freedom.
$\hat{\mathbf{I}}$	A 3×3 identity matrix.
$h^{(i)}$	A 3×1 Cartesian vector for link i. Defined as $h^{(i)} = \mathbf{m}_p \mathbf{P}_{p \to CG}^{(i)}$ it is only defined here to be used in a skew symmetric matrix $\tilde{h}^{(i)}$ form as a cross product operator to account for some components of the relationship angular acceleration contributions on joint p from of the linear acceleration of CG and vice-versa.
$\tilde{h}^{(i)}$	Skew symmetric form of $h^{(i)}$
$\mathbf{f}_p^{(i)}$	A 6×1 spatial vector containing an external force and moment applied about link i's joint p.
$\beta_p^{(i)}$	The bias force contribution from link i's rigid body.
$\beta_p^{*(i)}$	The bias force accumulated across the mechanism from the last link to link i.
$\mathbf{N}_p^{(i)}$	A 6×6 inertia tensor with the diagonal entry along joint degree of freedom stripped.
$\mathbf{k}_p^{(i)}$	An 6×1 array holding the column of the mass-inertia matrix $\mathbf{M}_p^{*(i)}$ associated with the joint degree of freedom, $\phi^{(i)}$.
$\zeta_p^{(i)}$	The velocity dependent acceleration terms, including Coriolis accelerations, described about link i's frame p; equivalent to $\zeta_d^{(i-1)}$.
$\zeta_d^{(i)}$	The velocity dependent acceleration terms, including Coriolis accelerations, described about link i's frame d; equivalent to $\zeta_p^{(i+1)}$.
$\zeta_\varepsilon^{(i)}$	The velocity dependent acceleration contribution from the actuation rate of link i's proximal joint.
$\tau_p^{(i)}$	The scalar force component acting in link i's joint degree of freedom.

$\tau_p^{*(i)}$	The scalar force component acting in link i's joint degree of freedom.
$\varepsilon_{target}^{(i)}$	Target joint displacement used by the PID controller.
$\dot{\varepsilon}_{target}^{(i)}$	Target joint displacement rate used by the PID controller.
$C_P^{(i)}$	The proportional coefficient for the PID controller of link i's proximal joint.
$C_I^{(i)}$	The integral coefficient for the PID controller of link i's proximal joint.
$C_D^{(i)}$	The damping coefficient for the PID controller of link i's proximal joint.
$P^{(i)}$	The power used to actuate link i's proximal joint.
$P_{max}^{(i)}$	The maximum allowable actuation power for link i's proximal joint, limits the joint controller.
$\tau_{PID}^{(i)}$	The load applied to link i's proximal joint by the controller.
$\tau_{PID,max}^{(i)}$	The maximum load applicable to link i's proximal joint based on a joint's displacement rate and maximum power $P_{max}^{(i)}$.
$\tau_{LL}^{(i)}$	The load applied to link i's proximal joint about its degree of freedom if a lower limit end stop is violated
$\tau_{UL}^{(i)}$	The load applied to link i's proximal joint about its degree of freedom if a upper limit end stop is violated
$\varepsilon_{LL}^{(i)}$	The lower limit end stop deflection of link i's proximal joint.
$\varepsilon_{UL}^{(i)}$	The upper limit end stop deflection of link i's proximal joint.

Contact Resolution Symbols

${}^B\mathbf{F}_{contact}$	The 6×1 spatial vector of force applied to the rigid body by the 3×1 contact force, expressed about the body-fixed frame.
${}^B\mathbf{T}_E$	The 6×6 transformation matrix used to transform a vector's frame of reference from the Earth-fixed frame to the body-fixed frame.

${}^{E}\mathbf{F}_{contact}$	The 6×1 spatial vector of force applied to the rigid body by the 3×1 contact force, expressed about the Earth-fixed frame.
${}^{E}\mathbf{f}_{contact}$	The 3×1 spatial vector containing the contact force.
${}^{E}\mathbf{P}_{B \to C_\sigma}$	A vector, starting at the body-fixed frame origin and ending at the centroid of pressure, expressed with respect to the Earth-fixed frame.
${}^{E}\mathbf{v}_{C_\sigma \mid A-B}$	A 6×1 spatial vector containing the relative velocity between objects A and B are the centroid of pressure, expressed with respect to the Earth-fixed frame.
${}^{E}\mathbf{T}_{B^{(A)}}$	The 6×6 transformation matrix used to transform a vector's frame of reference from the object A's body-fixed frame to the Earth-fixed frame.
${}^{E}\mathbf{T}_{B^{(B)}}$	The 6×6 transformation matrix used to transform a vector's frame of reference from the object B's body-fixed frame to the Earth-fixed frame.
${}^{B^{(A)}}\mathbf{T}^{(v)}_{B^{(A)} \to C_\sigma}$	The 6×6 velocity transformation matrix used to obtain the velocity at the centroid of pressure of object A given the velocity of the object at its body-fixed frame, expressed with respect to its body-fixed frame.
${}^{E}\mathbf{T}^{(v)}_{B^{(B)} \to C_\sigma}$	The 6×6 velocity transformation matrix used to obtain the velocity at the centroid of pressure of object B given the velocity of the object at its body-fixed frame, expressed with respect to its body-fixed frame.
${}^{B^{(A)}}\mathbf{v}_{B^{(A)}}$	The 6×1 spatial vector describing the absolute velocity of object A's body-fixed frame expressed with respect to object A's body-fixed frame.
${}^{B^{(B)}}\mathbf{v}_{B^{(B)}}$	The 6×1 spatial vector describing the absolute velocity of object B's body-fixed frame expressed with respect to object B's body-fixed frame.
$\mathbf{B}^{(A)}$	The body-fixed frame for object A.
$\mathbf{B}^{(B)}$	The body-fixed frame for object B.
C_σ	The centroid of pressure.
${}^{E}\mathbf{v}^{(\hat{\mathbf{t}})}_{C_\sigma \mid A-B}$	The component of ${}^{E}\mathbf{v}_{C_\sigma \mid A-B}$ that is tangential to contact normal direction $\hat{\mathbf{n}}$.

${}^{E}\mathbf{v}_{C_\sigma \mid A-B}^{(\hat{\mathbf{n}})}$	The component of ${}^{E}\mathbf{v}_{C_\sigma \mid A-B}$ that co-linear with contact normal direction $\hat{\mathbf{n}}$.
${}^{E}\mathbf{P}_{(min)}^{(A)}$	A 3×1 vector containing the Cartesian position on the surface of object A, representing one of the components of the minimum separation distance between objects A and B.
${}^{E}\mathbf{P}_{(min)}^{(B)}$	A 3×1 vector containing the Cartesian position on the surface of object B, representing one of the components of the minimum separation distance between objects A and B.
$f_{\hat{\mathbf{n}}}^{(s)}$	The stiffness component of the normal contact force.
$f_{\hat{\mathbf{n}}}^{(d)}$	The damping component of the normal contact force.
$f_{\hat{\mathbf{n}}}$	The normal contact force.
$\sigma_{\hat{\mathbf{n}}}(s)$	The normal local stress at a point s on the contact surface.
ds	The differential area used to define the local stress.
$e_{\hat{\mathbf{n}}}(s)$	The local engineering strain at a point s on the contact surface.
$\epsilon_{\hat{\mathbf{n}}}(s)$	The local true strain at a point s on the contact surface.
$\delta(s)$	The local deflection of the contact surface at a point s on the contact surface.
h	The spring-bed depth.
$\mathbf{P}_{E \to C\sigma}$	The 3×1 position vector starting at the origin of the Earth-fixed frame, ending at the centroid of pressure.
V	The displaced volumes of the spring.
V_{int}	The volume of interference.
$dV(s)$	The displaced volume of a differential spring.
E	Young's modulus.
E_A	Young's modulus for object 1.

E_B	Young's modulus for object 2.
h_A	The spring bed depth for object 1.
h_B	The spring bed depth for object 2.
$\dot{e}_{\mathbf{n}}(s)$	The local strain rate at some point s on the contact surface.
m_A	Strain-rate sensitivity exponent for object A.
m_B	Strain-rate sensitivity exponent for object B.
B_A	Damping factor B_1 of the material of object 1.
B_B	Damping factor B_1 of the material of object 2.
$e_r(s)$	The local radial strain at some point s on the contact surface.
$\delta R(s)$	The amount of local radial deformation at some point s on the contact surface.
R_i	The original undeformed radius.
$R_{eq}^{(A)}$	The equivalent radius for object A.
V_A	The volume of object A.
${}^{B^{(A)}}\mathbf{f}_{Coulomb}^{(A)(\hat{\mathbf{t}})}$	The Coulomb friction force applied to object A if there is relative motion to B
${}^{B^{(A)}}\mathbf{f}_{fric}^{(A)(\hat{\mathbf{t}})}$	The friction force applied to object A.
$f_{stick}^{(\hat{\mathbf{t}})}$	The maximum sticking friction force.
μ_c	The Coulomb friction coefficient.
μ_s	The sticking friction coefficient.
$f_{\hat{\mathbf{n}}}$	The normal contact force magnitude.

${}_{B^{(A)}}\mathbf{f}^{(A)}(\hat{\mathbf{t}})$	The tangential component of the forces acting on object A at its body-fixed frame, expressed about its body-fixed frame.
v_{tol}	The dead-zone tolerance velocity for the stick-slip transition.
V_{tetra}	The volume of a tetrahedron.
$\mathbf{v}_1$	The first vertex of a tetrahedron.
$\mathbf{v}_2$	The second vertex of a tetrahedron.
$\mathbf{v}_3$	The third vertex of a tetrahedron.
$\mathbf{v}_4$	The fourth vertex of a tetrahedron.
$\mathbf{P}_{centroid}$	The centroid location of a tetrahedron.
A	Object A.
B	Object B.
$\mathbf{P}_A$	The absolute position vector of object A.
$\mathbf{P}_B$	The absolute position vector of object B.
$\mathbf{P}_M^{(min)}$	The absolute position vector of the minimum separation distance on the surface of the Minkowski difference of objects A and B.
$\mathbf{P}_A^{(min)}$	The absolute position vector of the point on the surface of object A that represents one component of the minimum separation distance between objects A and B.
$\mathbf{P}_B^{(min)}$	The absolute position vector of the point on the surface of object B that represents one component of the minimum separation distance between objects A and B.
$S(A, \mathbf{d})$	A support mapping function, returns the furthest point on object A in the direction $\mathbf{d}$
$\mathbf{d}$	A search direction used to find the furthest points on an object.
$\mathbf{Q}$	A set of vertices, contains anywhere between 1 and 4 vertices at any one time.

$\mathbf{Q}_i$	The i^{th} vertex of in the set $\mathbf{Q}$.
c_i	The i^{th} barycentric coordinate.
$\mathbf{v}_i^{(A)}$	The vertex on object A corresponding to the i^{th} vertex in the set $\mathbf{Q}$.
$\mathbf{v}_i^{(B)}$	The vertex on object B corresponding to the i^{th} vertex in the set $\mathbf{Q}$.

This page intentionally left blank.

1 Introduction

Dynamic Systems Analysis (DSA) was contracted to construct an application programming interface (API) in the form of a C++ dynamic link library (DLL) that will enable DRDC Atlantic to perform high fidelity simulations of ship mechanical systems (SMS). The purpose of this report is to summarize the capabilities of the API and educate users on how to use the API to simulate a wide variety of scenarios.

There are four pieces of equipment that the SMS API is able to simulate: boom cranes, slack/taut cables, payloads (i.e. something hung from a winch cable) and winches. Figure 1 shows a scenario of how all four simulation object types could be simulated together. The SMS API was written using an object oriented approach in ANSI C++ such that within the API the dynamics of each item is encapsulated within four key classes[1]:

- `SMS::BoomCrane`
- `SMS::Cable`
- `SMS::Payload`
- `SMS::Winch`

Simulation of each of these components in a ship based scenario can be constructed within the SMS API by creating objects of these types, and calling the member functions of these objects.

This document outlines the theoretical background for each of these classes in the API, discusses the validation cases investigated, and describes practical usage of the API. Special attention was paid in the development of the API towards validation and fidelity of the software. This was accomplished by comparing simulation results with known analytical solutions or using published validation results.

In this report, first some further preliminaries on the basic usage of the SMS API are discussed, including definition of the SMS API coordinate system and an overview of the software architecture. Next, the theory behind each of the core models used in the key SMS API classes is discussed. Then, practical information on how to use the SMS API is presented. Lastly, a set of validation cases is presented which were used to test the API.

[1]The SMS API classes are contained within an `SMS` namespace so that the user of the API is clear which objects are defined within the SMS API DLL; this point will be emphasized throughout the report by fully qualifying the name of the class by prepending it with `SMS::`

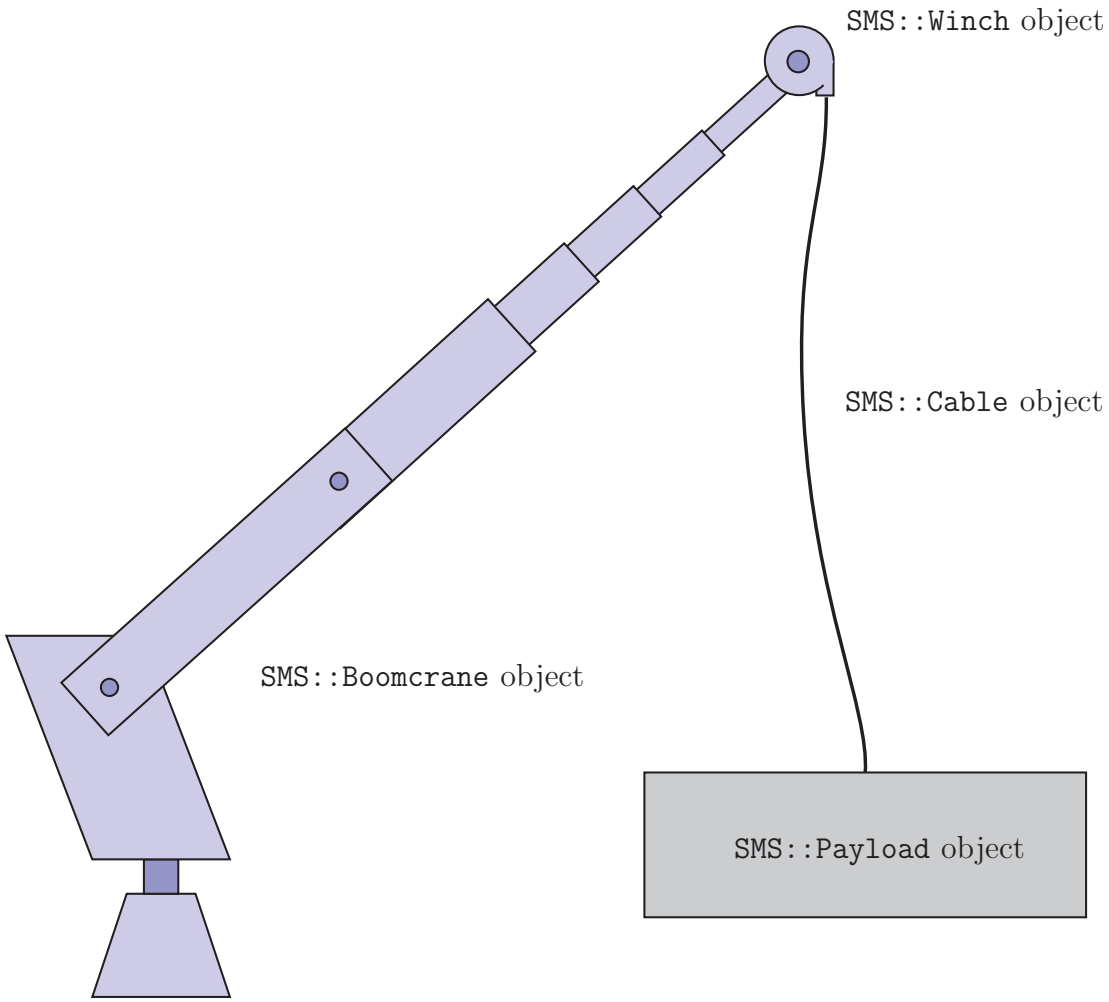


Figure 1: *Typical configuration whose simulation will be made possible with the classes available in the SMS API. The simulations will consist of cables, winches, cranes and payloads. Each of the classes must be able to interact.*

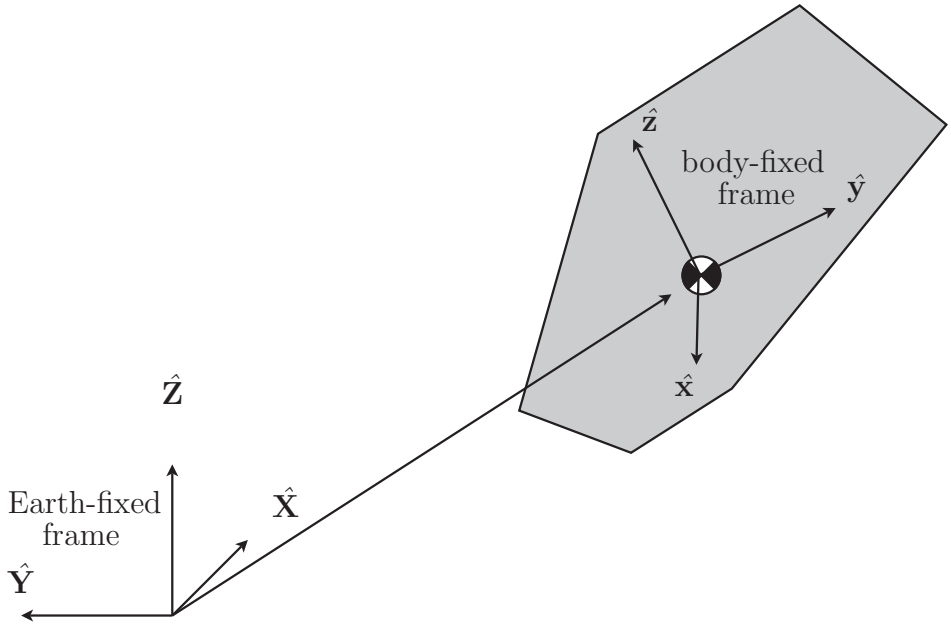


Figure 2: *The payload's Body-Fixed frame is defined relative to the Earth-Fixed Frame.*

1.1 Coordinate System

The `SMS::BoomCrane`, `SMS::Winch`, `SMS::Payload` and `SMS::Cable` classes represent physical objects that must be defined relative to a globally fixed coordinate system. To be consistent with other DRDC simulation programs, DSA has selected a coordinate system definition consistent with ShipMo3D. Thus, the SMS API has labeled the global system as an "Earth-fixed coordinate system". The Earth-fixed system is a right handed coordinate system that has its X axis pointing North and the Z axis point away from the center of the earth (+ Upward).

Local coordinate systems are attached to objects such as an `SMS::Payload`. They are defined in terms of their position and orientation relative to the Earth-Fixed Coordinate System (see Figure 2). The `SMS::RigidBody`'s local coordinate system is called as a "body-fixed coordinate system". This follows the convention of ShipMo3D.

1.2 SMS API Software Architecture

In the SMS API all key simulation object types, the `SMS::Cable`, `SMS::BoomCrane`, `SMS::Payload`, and `SMS::Winch`, are designed around the `SMS::SimObject` abstract base class, as seen in Figure 3. The `SMS::SimObject` class was designed to provide a

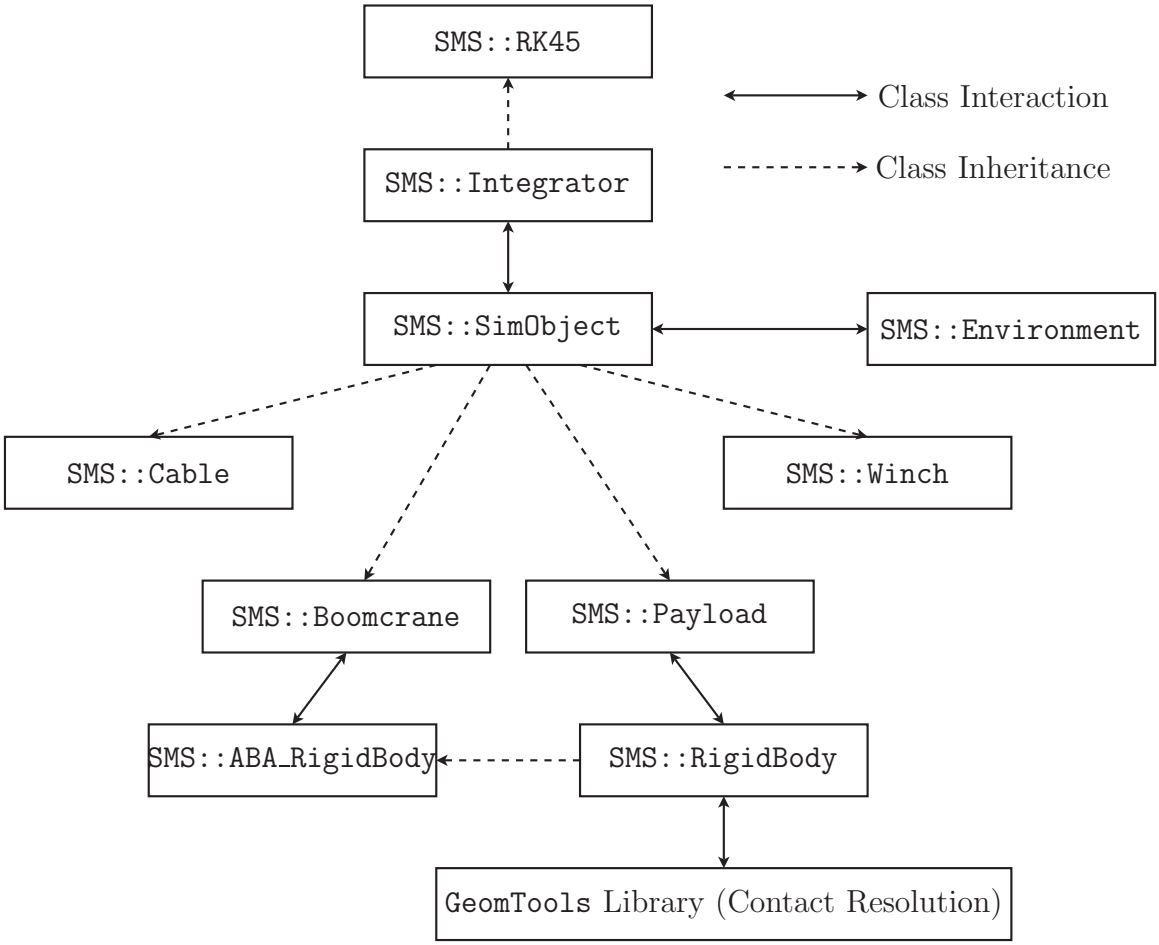


Figure 3: *The SMS API design architecture*

common interactive layer between simulation objects and the `SMS::Integrator` and `SMS::Environment` objects. The `SMS::Integrator` object is responsible for advancing the simulation time. The `SMS::Integrator` class polls each of the `SMS::SimObject` classes to calculate the dynamics of each `SMS::SimObject`. The integrator advances the simulation time by integrating the state vector derivatives over a time interval specified by the SMS API user. The `SMS::Environment` class is used by each `SMS::SimObject` to define key environmental parameters such as fluid velocity, density or sea state.

One thing to note from Figure 3 is that the `SMS::Payload` makes use of an `SMS::RigidBody` class to assist in the evaluation of its dynamics. In addition, the `SMS::Boomcrane` class also uses the `SMS::ABA_RigidBody` class to evaluate its dynamics. The SMS API user is never exposed to these utility classes, but they are fundamental to operation of the SMS API and are mentioned here for completeness.

Lastly, to simulate a wide variety of shipboard scenarios, it is necessary within the SMS API to account for contact effects between a payload and the ship deck. The `SMS::RigidBody` uses collision detection and contact resolution capabilities provided by the `GeomTools` library, which implements some computational geometry algorithms that allow collision detection and contact forces determination.

1.3 SimObject Interaction

How the `SMS::Cable`, `SMS::BoomCrane`, `SMS::Payload`, and `SMS::Winch` objects are connected to each other such that key information such as positions and forces can be passed between objects has been designed into each of the classes. Indeed, the SMS API has been designed to operate in two distinct ways:

1. as an encapsulated simulation where all mechanical systems objects are simulated on a single computer in a single executable[2]; and,

2. as part of a distributed simulation where some or all of the mechanical systems objects are simulated on different computers.

Each of these cases requires mechanisms such that boundary condition information can be passed between objects. In the first case, where all simulation objects are located within a single executable, the SMS API has been designed such that boundary condition data (i.e., forces from the cable on the boom crane, force from the payload on the cable, the winch payout speed, etc.) are shared between the simulation objects automatically. For instance, if an `SMS::Cable` object is connected to an `SMS::Payload` object, then the position of the connection point in space must be passed to the `SMS::Cable` object and the force exerted on the `SMS::Payload` from the `SMS::Cable` must be passed to the `SMS::Payload` object to evolve the coupled system dynamics. This automated feature has been designed into the SMS API to simplify setting up simulation scenarios. For simulations where simulation objects are distributed and integrated on separate computers, the SMS API user will have to manually pass this boundary condition information between attached simulation objects. Functions are provided to do this.

All `SMS::SimObject`s use their own `SMS::Integrator` object to integrate their dynamics through time. If however, objects are attached together with automated boundary condition updates, one object becomes the master and the others slaves. The master's integrator handles the integration of all attached objects by appending the state vectors of the slave objects, to its own state vector for integration. The `SMS::Integrator`, an abstract class itself, provides the core functionality that any integrator would require

[2]It should be noted that an encapsulated simulation could be part of a larger distributed simulation

to accomplish its job. Here, the SMS API uses the `SMS::RK45` integrator, which inherits `SMS::Integrator` and implements the Runge-Kutta 4(5) (Fehlberg) method, an adaptive numerical integrator that is used to solve the dynamics with a high degree of accuracy [3]. This integration method regulates the integration error by adjusting the integration time step which allows it to maintain a predetermined level of accuracy.

2 Theory

The following section outlines the theoretical background for each of the SMS API models implemented. First the finite-element based cable model that is used in the `SMS::Cable` class is reviewed. Second, the controller used to modify the cable length, that is encapsulated in the `SMS::Winch` class, is reviewed. Next, the `SMS::Payload` class's rigid body dynamics formulation is discussed. Next, the Articulated Body Algorithm, which is used by the `SMS::BoomCrane` class to resolve its dynamics, is presented. Lastly, the contact resolution scheme implemented is reviewed.

2.1 Cable

The purpose of the `SMS::Cable` class in the SMS API is to enable the simulation of towing cables, or any kind of flexible rope, wire, or chain, that may be suspended or towed from a ship in some fashion during a shipboard operation. For instance, it is possible that this model may be used to simulate at-sea replenishment or towed array operations. In general, this model should be capable of reporting internal cable tension and determining the path of the cable in space. Towards this end, a cubic-element lumped-mass, finite-element cable model was selected to evaluate the dynamics of such a cable. This model is presented in this section.

The finite-element model selected for use in the SMS API was originally developed by Dr. Brad Buckham at the University of Victoria [4, 5]. The benefits of the model are:

- the application of a cubic element to the finite element geometry allows more accurate results with longer element lengths than typical linear cable elements. This element choice allows element length increases of up to 50% over standard linear element methods [4, 6];
- utilizing a cubic spline to assist in cable dynamics evaluation significantly reduces the number of state variables needed to represent the line geometry, which increases simulation execution speed;
- including torsional and bending parameters allows accurate resolution of slack line and snap load events; and
- torsional mass inertia of the line itself is neglected and an instantaneous twist response results, which reduces high frequency torsional dynamics noise from inducing small numerical integration time steps and slowing execution speed.

To develop a finite element model capable of estimating the internal tensions of the cable and resolving snap loads and slack line events, consider the schematic in Figure 4.

In Figure 4, a low tension cable is connected between a boom crane and a generic payload where the cable path in space is defined in terms of global positions, $\mathbf{r}$. The cable is discretized with N nodes. The path of the cable in space is defined using a Frenet frame [7]. The local Frenet frame is defined at the nodal positions with the tangential vector $\hat{\mathbf{t}}$ and a normal direction $\hat{\mathbf{n}}$ that points to the center of curvature. The binormal vector $\hat{\mathbf{b}}$ completes the local Frenet coordinate system. To determine the degree of twist of the cable due to torsional deformation, a second local frame is defined that is permanently fixed to the cable's cross-section: $(\hat{\mathbf{q}}, \hat{\mathbf{p}}_1, \hat{\mathbf{p}}_2)$. It is separated from the Frenet frame by an angle of twist α.

The element state is defined by the node positions $\mathbf{r}^{(i)}$ in terms of the inertial coordinate frame, as shown in Figure 5. The curvature values, which are needed to calculate flexural response, are determined by fitting a cubic spline through the node points. The cubic spline kinematically enforces continuity of slope across the span of the cable as a function of node positions. The time evolution of the cable state is found by determining the nodal linear accelerations from Newton's second law; a numerical integration algorithm is used to evolve the resulting node positions through time.

The internal reaction forces required for Newton's second law are found by considering a differential segment of cable. Internal reaction forces due to bending (κ), axial (ϵ), and torsion (τ) strains can be resolved with assistance of the Frenet frame after some manipulation as detailed in [4]. The cubic spline curvatures at the node points are used to establish bending strain. Axial element strains are calculated from the straight-line distance between nodes and knowledge of the unstretched element lengths. Torsional deformations are quantified with the knowledge of the Frenet frame torsion and the assumption of negligible torsional inertia that provides a method to calculate the separation between mechanical frame and Frenet frame (by angle α) at the node points. Applying the Galerkin criterion transforms the continuous equations for internal reaction and externally applied distributed forces into a discrete form that can be implemented numerically in a straightforward manner using linear algebra. Some discussion of how the Galerkin criterion is applied is provided in Appendix C. Before the Galerkin criterion can be applied, a trial solution to the system, which incorporates the shape functions that blend the discrete node values of the mesh into a continuum, must be selected. The trial solution for the continuous cable position, $\mathbf{r}(s)$, uses cubic shape functions and the positions and curvatures of the node points at the boundary of the element:

$$\mathbf{r}(s) = \mathbf{r}^{(0)}\phi_{1,1} + \mathbf{r}''^{(0)}\phi_{1,2} + \mathbf{r}^{(1)}\phi_{1,3} + \mathbf{r}''^{(1)}\phi_{1,4} \tag{1}$$

where the first element has four shape functions, $\phi_{i,j}$, $j = 1, 2, 3, 4$, $i = 1, 2, \cdots N$:

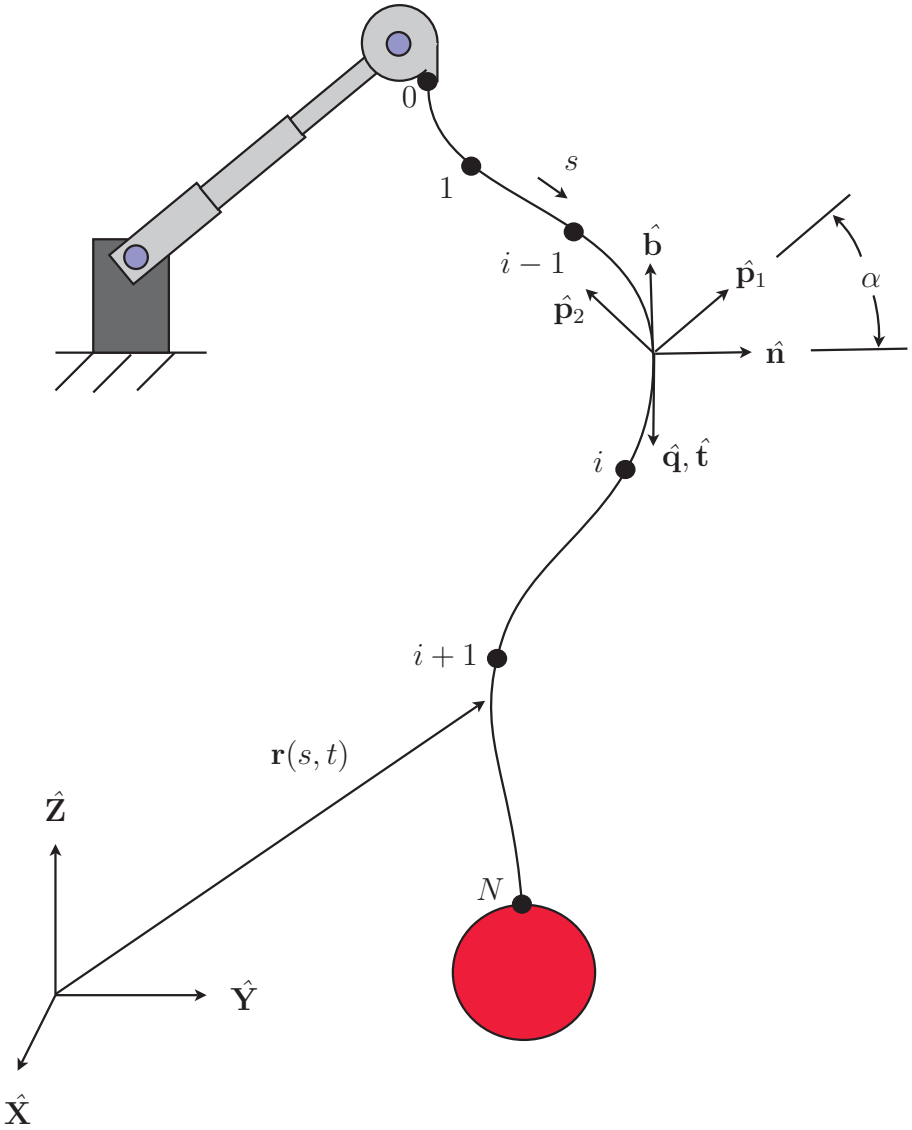


Figure 4: *Schematic of a slack cable model developed in terms of a frame attached to the cable cross section and Frenet frame*

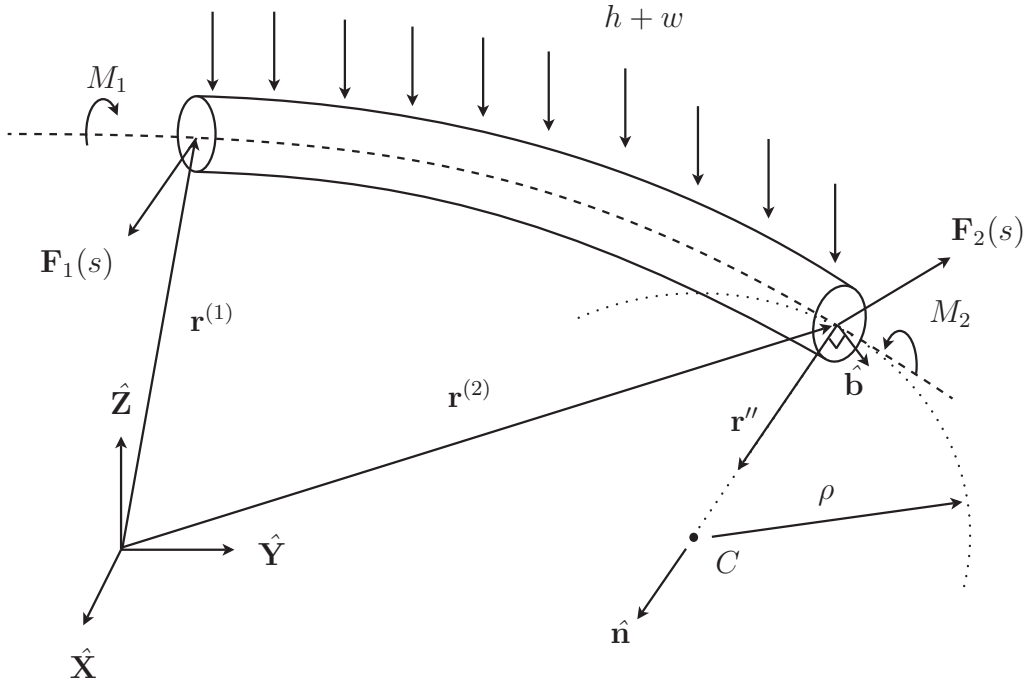


Figure 5: *The figure shows a single cable element with the distributed gravitational and hydrodynamic forces w and h, respectively.*

$$\begin{aligned}
\phi_{1,1} =& \frac{s^{(1)} - s}{L_u^{(1)}} \\
\phi_{1,2} =& \frac{1}{6}\left(\phi_{1,1}^3 - \phi_{1,1}\right)(L_u^{(1)})^2 \\
\phi_{1,3} =& \frac{s - s^{(0)}}{L_u^{(1)}} \\
\phi_{1,4} =& \frac{1}{6}\left(\phi_{1,3}^3 - \phi_{1,3}\right)(L_u^{(1)})^2
\end{aligned}$$

Note the first subscript indicates the element number. The equation for an element produced from using these shape functions in conjunction with the Galerkin criterion produces the discrete cable element dynamics equations:

$$\left(\mathbf{K}_\epsilon + \mathbf{K}_\kappa + \mathbf{K}_\tau\right)\mathbf{X} + \mathbf{W} + \mathbf{H} = \mathbf{M}_C\ddot{\mathbf{X}} \tag{2}$$

where $\mathbf{X}$ is the assembly of node positions $\mathbf{r}^{(i)}$ and curvatures $\mathbf{r}''^{(i)}$. The full form of the mass and stiffness matrices for the consistent cable, as well as sample concatenated

matrices for a system of two elements, can be seen in Appendix A. The first element is bounded by nodes $i = 0, 1$, which results in a length 12 vector:

$$\mathbf{X} = \{\mathbf{r}^{(0)T}\ \mathbf{r}''^{(0)T}\ \mathbf{r}^{(1)T}\ \mathbf{r}''^{(1)T}\}^T \tag{3}$$

The length 12 force vector $\mathbf{W}$ is combined weight and buoyancy and $\mathbf{H}$ is hydrodynamic load in terms of the global inertial reference frame. $\mathbf{M}_C$ is the 12×12 consistent element mass matrix. $\mathbf{K}_\epsilon$, $\mathbf{K}_\kappa$ and $\mathbf{K}_\tau$ are the axial, bending, and torsional stiffness 12×12 matrices.

In order to simplify the system of equations for computational and model efficiency, a lumped mass assumption is made. A direct result of lumping the mass at the element nodes is that no curvature inertia is available to couple forces to curvature accelerations; these terms are dropped from equation 2 to form the reduced cable element dynamics equation:

$$(\mathbf{K}_{\epsilon\ G} + \mathbf{K}_{\kappa\ G} + \mathbf{K}_{\tau\ G})\ \{\mathbf{r}^{(0)T}\ \mathbf{r}''^{(0)T}\ \mathbf{r}^{(1)T}\ \mathbf{r}''^{(1)T}\}^T + \mathbf{W}_G + \mathbf{H}_G = \mathbf{M}_G \ddot{\mathbf{R}}_G \tag{4}$$

where the length 6 force vector $\mathbf{W}_G$ is combined weight and buoyancy and $\mathbf{H}_G$ is hydrodynamic load in terms of the global inertial reference frame. Finally, $\mathbf{K}_{\epsilon\ G}$, $\mathbf{K}_{\kappa\ G}$ and $\mathbf{K}_{\tau\ G}$ are the axial, bending, and torsional stiffness 6×12 matrices. Finally, as the curvatures are a function of node positions via the cubic spline, the reduced cable state can be written in terms of node positions:

$$\mathbf{R}_G = \{\mathbf{r}^{(0)T}\mathbf{r}^{(1)T}\}^T \tag{5}$$

In place of dynamic curvature information, kinematic curvature information is provided by fitting a cubic spline through the cable node points. The cubic spline equations, which are solely a function of the node positions and termination type (clamped or pinned), enforce continuous slope and curvature across the concatenated system of elements that form the entire cable. The lumped mass model has substantially reduced computational overhead as the cable state vector is reduced by half. In addition, the block-diagonal nature of the lumped mass matrix means that 3×3 systems of equations are solved simultaneously instead of the minimum block diagonal 12×12 with the consistent mass matrix. The full form of the mass and stiffness matrices for lumped cable, as well as sample concatenated matrices for a system of two elements, can be seen in Appendix A.

As mentioned previously, the total mechanical torsion, τ, is found from the Frenet frame torsion, γ, which is a direct function of node positions, and with assumption of negligible torsional inertia that guarantees a uniform mechanical torsional deformation along the entire cable length at any point in time. The mechanical torsion of the element is a function of the Frenet frame torsion and the angle between the mechanical frame and Frenet frame, α:

$$\tau = \gamma + \frac{d\alpha}{ds} \tag{6}$$

With the assumption of negligible element torsional inertia, the resulting torsion deformation of the entire cable must be uniform:

$$\frac{d\left(GJ\tau\right)}{ds} = 0 \tag{7}$$

The resulting discrete equations produced from applying the Galerkin criterion are:

$$GJ\mathbf{L}_G\alpha_G = GJ\{\Delta\gamma_G\} \tag{8}$$

where $\mathbf{L}_G$ is a tridiagonal matrix based on the element lengths, γ_G is a vector of Frenet frame torsions measured at the node points, and α_G is the vector of torsional deformation from the Frenet frame to the mechanical cable frame. Since the torsion of the Frenet frame is a known function of the cable node positions, solving the set of equations for α_G produces the total mechanical torsion of the cable at any instance in time. Note that the matrix $\mathbf{L}_G$ is singular and requires boundary α values to be specified for the system of equations to be solved. This is a physical consequence from assuming negligible torsional inertia: if either or both ends of the cable are torsionally unconstrained, an infinite number of solutions exist for the cable torsion for any applied torques. As such, once boundary α values are obtained, the system torsion can be resolved from equation 8.

In the model implemented, torsion evolves through constraining the boundary mechanical frames, either via controller or by a rigid body connection, which is a kinematic constraint that indirectly specifies the boundary α values. However, resolving the boundary α values is complicated by the fact that the Frenet frame torsion, while a function of the geometry of the cable, particularly the normal direction, is not defined in areas of no curvature, or at inflection points [3]. The α value at the first node of the cable, α_0,

[3] An inflection point is a point on the Cable where the curvature vector changes direction by 180 degrees.

is quantified from the angular separation about the tangent between the Frenet frame and the kinematically specified mechanical frame. If no curvature exists at that boundary (i.e. below a discrete threshold), the binormal of the Frenet frame at the nearest downstream point that has nonzero curvature is used at the boundary without loss of generality. If the cable is completely straight, no Frenet frame can be defined and the boundary α value is zero; this effectively constrains the Frenet frame uniformally to the mechanical frame at node 0. Next, the α value at the other end of the cable, at the node N boundary, must be quantified.

However, quantifying boundary α values must take into account the effects of temporally evolving inflection points. Inflection points that exist along the span of the cable can introduce a maximum spatially instantaneous Frenet frame rotation of π radians and evolve naturally through the course of almost any simulation. Since a discrete curvature tolerance is used to indicate a straight tether, an instantaneous Frenet frame torsion could occur on the next dynamics evaluation and cause difficulties with numerical integration and introduce artificial torsion impulses. An example of how such inflections can introduce an artificial torsion load is with a planar cable with identically orientated mechanical frames at the boundaries, with one inflection point. The Frenet frame torsion of π occurs at the inflection point, resulting in a flip of the Frenet normal and binormal direction. Quantifying the α_N value in an analogous manner to α_0 for this case would lead to $\alpha_N = \pi$ despite the fact that the tether profile is planar and no mechanical twist has been induced by boundary mechanical frame movement. Without compensating for the presence of inflection points, artificial mechanical torsions will result when the discrete Equations 8 are resolved. Inflection points that temporally evolve are successfully compensated for with the use of past twist information.

Before quantifying α_N, the influence of any inflection point at node 0 is compensated for by simply quantifying benchmark values for α_0 that are offset by increments of π; the new α_0 value selected has a minimum difference between the last physical time step value. Note that since only the spatial gradient of α is needed to quantify total torsion in Equations 8, applying a compensating offset value in multiples of π will not affect the mechanical torsion calculated.

With the consistent α_0 value quantified, α_N is computed in a similar manner with a slight variation. The value of α_N must be consistent with α_0 and the past mechanical twist state such that there is no extreme discontinuity in twist induced by new temporally introduced inflection points. The initial estimate of α_N is simply the value of α_0 plus the spatial integral over the span of the cable of the past mechanical twist of each element (uniform) with the present Frenet frame torsion. This produces an estimate of what α_N should be, α_N^*, that incorporates the offset value introduced by a possible inflection at the first node, the expected position due to past (last physical time step) twist state, and the behavior of the Frenet frame along the span of the cable. As with

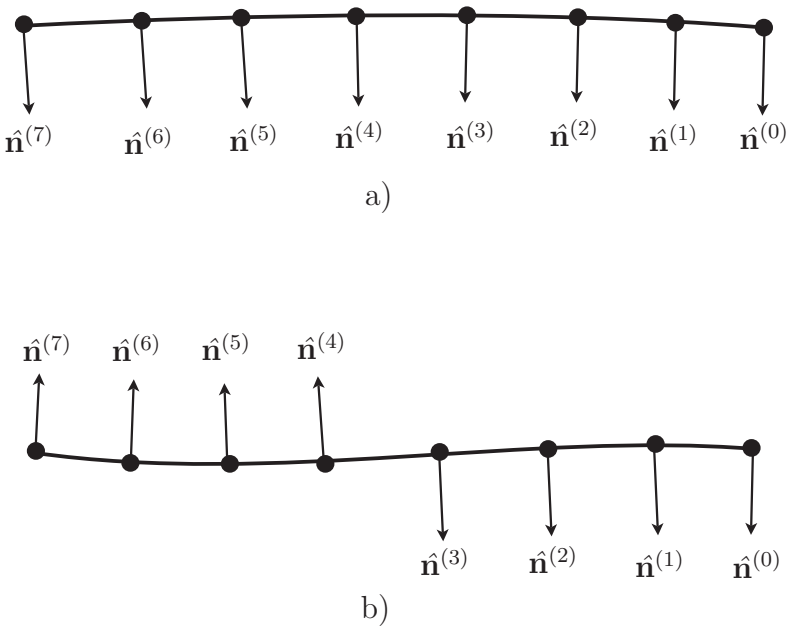


Figure 6: *a) Before temporal inflection: curvature and Frenet frame are well defined everywhere. b) After temporal inflection: inflection between nodes 3 and 4 introduces a Frenet frame twist of π radians and must be compensated for in torsion calculations. Note that the difference between states a) and b) can be merely one dynamics evaluation.*

α_0, the initial estimate of α_N is produced from the angle between the Frenet frame and mechanical frame axis at node N. The initial value of α_N is offset by increments of π such that the difference between α_N and the estimate α_N^* is minimized within a tolerance of $\frac{\pi}{2}$.

In the planar cable example mentioned previously, this algorithm detects the instantaneous increase by π due to the temporally evolved inflection and removes the effect from the boundary α estimate. This is a requirement to handle the many inflections that evolve and dissipate through a numerical simulation, especially in low tension events. With the boundary α values computed, the mechanical torsion is quantified and used in the torsion stiffness matrices to compute the resulting force on the cable nodes. Sample matrix equations of the torsional stiffness matrix are provided in Appendix B.

Note that the principle cable properties that need to be identified through experiment or extrapolated from manufacturer data sheets are:

- EI (Nm^2) bending stiffnesses
- GJ (Nm^2) torsional stiffness
- EA (N) axial stiffness

For synthetic lines and chains, the bending and torsional stiffnesses are often negligible and axial data can be readily obtained. However, for simulations involving wire rope or armored communications cables in low tension situations, bending and torsional effects can be significant and should be calculated or measured. DSA has a compiled list of stiffness numbers that can be used to test the cable classes functionality and interaction with other DRDC Atlantic federates or classes.

2.2 Winch

The purpose of the `SMS::Winch` class is to simulate the behaviour of a shipboard winch that is used to render or recover wire or rope at some rate. It is possible, that a winch will be designed to maintain a constant line tension. This section reviews how these winch behaviors are simulated in the SMS API.

A basic winch model was developed for the `SMS::Winch` class. Neither the drum or gearbox is modeled; the `SMS::Winch` simply serves as a controller for the `SMS::Cable` class. The `SMS::Winch` class has the ability to lengthen or shorten the boundary element of the end of the `SMS::Cable` it is attached to through setting payout or payin rates, which are the derivative of the element length. While this is a particularly simple model, it can serve as a basic testbed tool as well as a foundation upon which more complex

winch models can be constructed. The `SMS::Winch` class was designed to operate in constant tension and constant velocity control modes.

The winch changes the length of cable in the simulation by changing the length of the element that the winch is attached to. The cable element lengths are controlled by setting minimum or maximum length limits. If the maximum is exceeded, a node is inserted at the midpoint of the element adjacent to the winch. If the element length becomes less than the minimum limit, a node is removed and the boundary element joins the element adjacent to it.

When changing the length of the boundary elements, the resulting payout forces due to the mass flux of the cable are applied on the first interior node. To determine the length of the boundary elements throughout the simulation, the rate of change of the boundary element lengths (the payout or payin rate) are numerically integrated along with the other cable state variables.

An `SMS::Winch` object can only be attached to either node 0 or node N of a cable. The `SMS::Winch` object has two settings, a constant tension mode as well as a constant velocity mode. The rate at which the payout rate changes is governed by the acceleration and deceleration settings, such that the winch speed v^w changes over an integration timestep of length ΔT (from time t_{k-1} to t_k) according to:

$$v^w_{t_k} = a^w_{accel} dt + v^w_{t_{k-1}} \quad , if, \; v^w < v^w_{set} \tag{9}$$

and

$$v^w_{t_k} = a^w_{decel} dt + v^w_{t_{k-1}} \quad , if, \; v^w > v^w_{set} \tag{10}$$

where the winch deceleration and acceleration settings are $a^w_{decel} < 0$ and $a^w_{accel} > 0$, respectively. The velocity setpoint is v^w_{set}. If the winch velocity is above or below the setpoint velocity, the winch will adjust its velocity to reach the setpoint using the acceleration and deceleration parameters. A slowly responding winch (one with high inertia), can be simulated by decreasing the acceleration and deceleration parameters. The deceleration parameter can be adjusted to account for braking effects.

To achieve constant tension control in the winch, PID gains are used to set the payout and payin velocity for the winch based on the tension in the cable at the boundary elements. For instance, if a constant tension mode winch is attached to node N in the cable, the winch will respond to changes in tension in element N of the cable. If the constant tension mode winch is attached to node 0, the winch will respond to changes in tension in element 1 of the cable.

2.3 Payload

The purpose of the `SMS::Payload` is to simulate the motion of some "payload". For example, a payload, such as a small rigid-hull inflatable rescue craft, may be suspended from a cable and then dropped overboard or onto the deck of the ship. The `SMS::Payload` class is designed to determine the motions of the payload in response to these actions. The payload dynamics formulation is presented in this section.

In the SMS API a payload is treated as a rigid body, which responds to forces according to the Newton-Euler equations of motion. The Newton-Euler equations of motion state that the relationship between the acceleration of the center of gravity of a 6 degree of freedom (DOF) rigid body and a force applied to it is:

$$^{E}\mathbf{F} =^{E} \mathbf{M}^{E}\mathbf{a}_{cg} \tag{11}$$

where $^{E}\mathbf{F}$ is a 6 DOF spatial vector that contains the three linear and three angular force components $\{^{E}F_x, ^{E}F_y, ^{E}F_z, ^{E}\tau_x, ^{E}\tau_y, ^{E}\tau_z\}$ and $^{E}\mathbf{a}_{cg}$ is the spatial acceleration vector that includes the three linear and three angular acceleration components $\{^{E}a_x, ^{E}a_y, ^{E}a_z, ^{E}\alpha_x, ^{E}\alpha_y, ^{E}\alpha_z\}$. The inertia matrix of a rigid body whose body-fixed frame is aligned with and located at the origin of the Earth-fixed frame is:

$$^{E}\mathbf{M} = \begin{bmatrix} m & 0 & 0 & 0 & 0 & 0 \\ 0 & m & 0 & 0 & 0 & 0 \\ 0 & 0 & m & 0 & 0 & 0 \\ 0 & 0 & 0 & I_{xx} & I_{xy} & I_{xz} \\ 0 & 0 & 0 & I_{yx} & I_{yy} & I_{yz} \\ 0 & 0 & 0 & I_{zx} & I_{zy} & I_{zz} \end{bmatrix} \tag{12}$$

where m is the rigid body mass, I_{xx}, I_{yy}, and I_{zz} are the mass moments of inertia about the inertial frame axis, and I_{xy}, I_{xz}, and I_{yz} are the associated products of inertia.

The Newton-Euler equations of motion can be derived from the total rate of change of momentum, that is, the Newton-Euler equations are derived from:

$$^{E}\mathbf{F} = \frac{d}{dt}(^{E}\mathbf{M}^{E}\mathbf{v}_{cg}) \tag{13}$$

where $^{E}\mathbf{v}_{cg}$ is the velocity of the rigid body in Plücker coordinates.

The Newton-Euler equations of motion describe the motion of a rigid body observed from a fixed inertial frame. To facilitate simulation, it is convenient to evaluate the dynamics with respect to a frame that is fixed to the rigid body. An advantage of this method is that the spatial inertia matrix, as well as other information like cable connection location, remains constant when expressed with respect to a body-fixed reference frame.

2.3.1 The Equations of Motion of a Rigid Body about the Body-Fixed Frame

To develop the equations of motion of the rigid body with respect to the body fixed frame, the position and orientation of the rigid body with respect to the Earth-fixed frame E is first defined as ${}^{E}\check{\mathbf{P}}_{cg} = [{}^{E}P_x, {}^{E}P_y, {}^{E}P_z, \phi, \theta, \psi]^T$ as shown in Figure 7. The 6×1 array ${}^{E}\check{\mathbf{P}}_{cg}$, contains three Cartesian space position components and a $Z(\psi) - Y'(\theta) - X''(\phi)$ body-fixed Euler angle sequence set, where $Z(\psi)$ represents a rotation about the Z axis of Earth-fixed inertial coordinate system, $Y'(\theta)$ is the rotation about the intermediate Y' axis, and $X''(\phi)$ is the rotation about the final X'' axis. Taking the derivative of ${}^{E}\check{\mathbf{P}}_{cg}$ yields:

$$ {}^{E}\check{\mathbf{v}}_{cg} = {}^{E}\dot{\check{\mathbf{p}}}_{cg} \tag{14} $$

where ${}^{E}\check{\mathbf{v}}_{cg} = [{}^{E}\dot{P}_x, {}^{E}\dot{P}_y, {}^{E}\dot{P}_z, \dot{\phi}, \dot{\theta}, \dot{\psi}]^T$ is a 6×1 array that defines both the linear velocity of the rigid body's center of gravity expressed in relation to the inertial frame, E, and the rate of change of the body fixed coordinate system's orientation or angular velocity. A special transformation matrix ${}^{E}\mathbf{T}_B$ is required to map the spatial velocity of the body in terms of the body frame to the non-Plücker Euler angle rate based velocity array, ${}^{E}\check{\mathbf{v}}_{cg}$:

$$ {}^{E}\check{\mathbf{v}}_{cg} = {}^{E}\mathbf{T}_B \, {}^{B}\mathbf{v}_{cg} \tag{15} $$

where ${}^{B}\mathbf{v}_{cg} = \{u, v, w, p, q, r\}$ is a 6 DOF spatial velocity vector formed with the linear velocity vector components u, v, w and angular velocity vector components p, q, r of the rigid body described with respect to the body-fixed frame; ${}^{E}\mathbf{T}_B$ is the 6×6 transformation matrix that maps the linear velocity component from the body-fixed frame to the inertial frame and maps the angular velocity component from the body-fixed frame to the Euler angle rates as given in [8]:

$$ {}^{E}\mathbf{T}_B = \begin{bmatrix} \mathbf{R}_1 & 0_{3\times3} \\ 0_{3\times3} & \mathbf{R}_2 \end{bmatrix} \tag{16} $$

Since the orientation of the body-fixed frame with respect to the inertial frame is described using the $\mathbf{R}_{ZY'X''}$ Euler angle set, $\mathbf{R}_1$ is [8]:

$$ \mathbf{R}_1 = \begin{bmatrix} c\phi c\theta & c\phi s\theta s\psi - s\phi c\psi & s\phi s\theta c\psi + s\phi s\psi \\ s\phi c\theta & +s\phi s\theta s\psi + c\phi c\psi & s\phi s\theta c\psi - c\phi s\psi \\ -s\theta & c\theta s\psi & c\theta c\psi \end{bmatrix} \tag{17} $$

where $c\chi$, $s\chi$, and $t\chi$ are the sine, cosine, and tangent of an angle χ, and ϕ, θ, ψ are the roll, pitch, and yaw angles. The 3×3 transformation matrix $\mathbf{R}_2$, which maps body-fixed

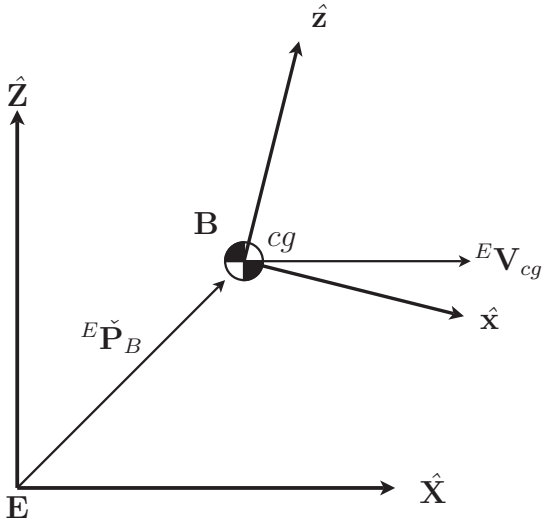


Figure 7: *Frame B described with respect with to frame E*

frame angular velocity to the non-orthogonal Euler angle rates, is:

$$\mathbf{R}_2 = \begin{bmatrix} 1 & s\phi t\theta & c\phi t\theta \\ 0 & c\phi & -s\phi \\ 0 & s\phi c\theta & c\phi/c\theta \end{bmatrix} \tag{18}$$

Taking the time derivative of ${}^{B}\mathbf{v}_{cg}$ will yield the acceleration ${}^{B}\mathbf{a}_{cg}$. Since frame B is a moving and rotating frame, the total time rate of change of the velocity vector is:

$${}^{B}\mathbf{a}_{cg} = {}^{B}\dot{\mathbf{v}}_{cg} + {}^{B}\boldsymbol{\Omega}_{B}^{B}\mathbf{v}_{cg} \tag{19}$$

where ${}^{B}\mathbf{a}_{cg}$ is the absolute acceleration vector of the rigid body in terms of the body-fixed frame B and accounts for all rates of change and the effect of the rotating reference frame, and ${}^{B}\boldsymbol{\Omega}_{B}$ is:

$${}^{B}\boldsymbol{\Omega}_{B} = \begin{vmatrix} 0 & -\omega_z & \omega_y & 0 & 0 & 0 \\ \omega_z & 0 & -\omega_x & 0 & 0 & 0 \\ -\omega_y & \omega_x & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -\omega_z & \omega_y \\ 0 & 0 & 0 & \omega_z & 0 & -\omega_x \\ 0 & 0 & 0 & -\omega_y & \omega_x & 0 \end{vmatrix}$$

The Newton-Euler equations expressed with respect to frame B become:

$${}^{B}\mathbf{F} = \mathbf{M}{}^{B}\mathbf{a}_{cg} \tag{20}$$

Substituting equation 19 into 20 yields:

$$ {}^{B}\mathbf{F} = \mathbf{M}{}^{B}\dot{\mathbf{v}}_{cg} + {}^{B}\,\boldsymbol{\Omega}_{B}\mathbf{M}{}^{B}\mathbf{v}_{cg} \tag{21} $$

To evolve the dynamics of the rigid body requires solving equation 21 for ${}^{B}\dot{\mathbf{v}}_{cg}$. To do this, the second term on the right hand side of equation 21, now called F_c, is a bias force that accounts for apparent accelerations resulting from the angular rotation of the frame:

$$ \mathbf{F}_{c} = {}^{B}\,\boldsymbol{\Omega}_{B}\mathbf{M}{}^{B}\mathbf{v}_{cg} \tag{22} $$

such that the equation of motion is written as:

$$ {}^{B}\dot{\mathbf{v}}_{cg} = \mathbf{M}^{-1}({}^{B}\mathbf{F} - \mathbf{F}_{c}) \tag{23} $$

Numerical integration of ${}^{B}\dot{\mathbf{v}}_{cg}$ will yield results that are not complete since frame B is moving with the center of gravity of the rigid body. To properly integrate the dynamics and simulate the behavior of the payload, ${}^{B}\dot{\mathbf{v}}_{cg}$ must be expressed with respect to the fixed inertial reference frame E. The acceleration of the rigid body in terms of frame E, the 6×1 array ${}^{E}\breve{\mathbf{a}}_{cg}$, is derived in terms of the inertial frame through differentiation of equation 15 as presented in [8]:

$$ {}^{E}\breve{\mathbf{a}}_{cg} = {}^{E}\,\mathbf{T}_{B}\;{}^{B}\dot{\mathbf{v}}_{cg} + {}^{E}\,\dot{\mathbf{T}}_{B}\;{}^{B}\mathbf{v}_{cg} \tag{24} $$

where the rate of change of spatial transformation matrix ${}^{E}\dot{\mathbf{T}}_{B}$ is obtained as [9]:

$$ {}^{E}\dot{\mathbf{T}}_{B} = \begin{vmatrix} \dot{\mathbf{R}}_{1} & 0_{3\times3} \\ 0_{3\times3} & \dot{\mathbf{R}}_{2} \end{vmatrix} \tag{25} $$

where $\dot{\mathbf{R}}_{1}$ is:

$$ \dot{\mathbf{R}}_{1} = \boldsymbol{\Omega}\mathbf{R}_{1} \tag{26} $$

and because the rotational components of ${}^{E}\breve{\mathbf{a}}_{cg}$ are described about the axes of frame resolved through the non-orthogonal Euler angle rotation sequence, $\dot{\mathbf{R}}_{2}$ is:

$$ \dot{\mathbf{R}}_{2} = \begin{vmatrix} 0 & c\phi\dot{\phi}t\theta + (1+t^{2}\theta)s\phi\dot{\theta} & -s\phi\dot{\phi}t\theta \\ 0 & -s\phi\dot{\phi} & -c\phi\dot{\phi} \\ 0 & (c\phi c\theta\dot{\phi} + s\phi s\theta\dot{\theta})/c^{2}\theta & (-s\phi c\theta\dot{\phi} + c\phi s\theta\dot{\theta})/c^{2}\theta \end{vmatrix} \tag{27} $$

and $\mathbf{\Omega}$ is a 3×3 skew symmetric matrix of the body rotation rates. Now that the accelerations are quantified in terms of the inertial Earth-fixed frame, the dynamics can be evolved through time with a numerical integrator.

In the SMS API, the `SMS::Payload` class uses the `SMS::RigidBody` class to evaluate dynamics calculations presented above. The state and state derivatives provided by the `SMS::RigidBody` are integrated through time by the `SMS::Payload`'s integrator to evolve the dynamics through time. The `SMS::Payload` was designed in this fashion because the `SMS::Boomcrane`, discussed in Section 2.4.2.1, uses a modified `SMS::RigidBody` with some added functionality for the dynamics evaluation of its links. The following description of the dynamics of a rigid body were encapsulated within in the `SMS::RigidBody` class.

2.3.2 Cable-Payload Interactions

When an `SMS::Cable` is attached, it imparts an external force onto the `SMS::Payload`. The force that the `SMS::Cable` imparts on the `SMS::Payload` is the result of its elastic deformations. Tension, the result of axial strain, is the linear force component while pure moments are produced from flexural and torsional strains. If a swivelling pin joint is used, the torsional and bending strains are always zero at the boundary of the cable. The connection point with respect to the body-fixed frame, ${}^{B}\mathbf{P}_{A}$, which is constant through simulation, must be provided (see figure 8) while a clamped connection requires the cable mechanical frame be defined by the principle vectors $\mathbf{P}_1$ and $\mathbf{P}_2$, also in terms of the body-fixed frame. Note that the cross product of these principle vectors produces the tangent direction at that end of the cable.

The cable provides a spatial force vector, ${}^{E}\mathbf{F}_{Cable}$, described with respect to the Earth-fixed frame, which is provided to the rigid body for its dynamics evaluation while the payload defines the position of the end of the cable to which it is attached.

2.4 BoomCrane

The purpose of the articulated boom crane model and the `SMS::BoomCrane` class is to simulate the motion of a shipboard boom crane, like the one shown in Figure 9, during typical operations. The model may be used to determine the operability of such equipment in various sea-states, or may be used to assess the feasibility of certain crane operations. To simulate a crane, the SMS API relies on a robotics algorithm called the Articulated Body Algorithm. This, and the general implementation of the crane simulator inside the SMS API, is presented below.

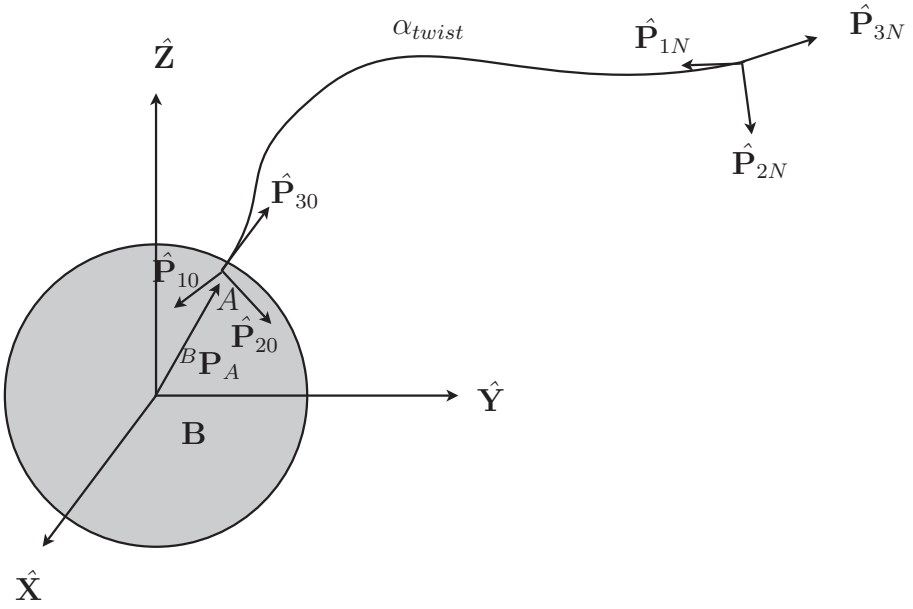


Figure 8: *The mechanical frame of a cable attached at a point ${}^{B}\mathbf{p}_{A}$ relative to the rigid body.*

2.4.1 Articulated Bodies Overview

To perform dynamic simulation of the crane motion, it is assumed the motion of the base of the crane and the disturbance forces on the crane's links are known. The disturbance forces may be from cables and payloads hanging from the boom crane, wind loads on the crane structure, or any other force acting on one of the links in the crane. The analysis of the motion of any articulated mechanism, such as a boom crane, is classified as one of two types of problems:

- The inverse dynamics problem, which calculates the joint actuation forces required to achieve an assumed mechanism motion.
- The forward dynamics problem, which calculates the mechanism motion through time given the joint actuation forces and external disturbance forces on the mechanism.

To dynamically simulate the motion of a boom crane, the forward dynamics problem must be solved. More specifically, the forward dynamics problem solves for the joint accelerations of the mechanism. The joint accelerations are calculated using the known inertia and mass of the links in the system as well as the actuation forces and external forces on the system. Once the joint accelerations are determined, they can be integrated through time to quantify the link velocities and positions.

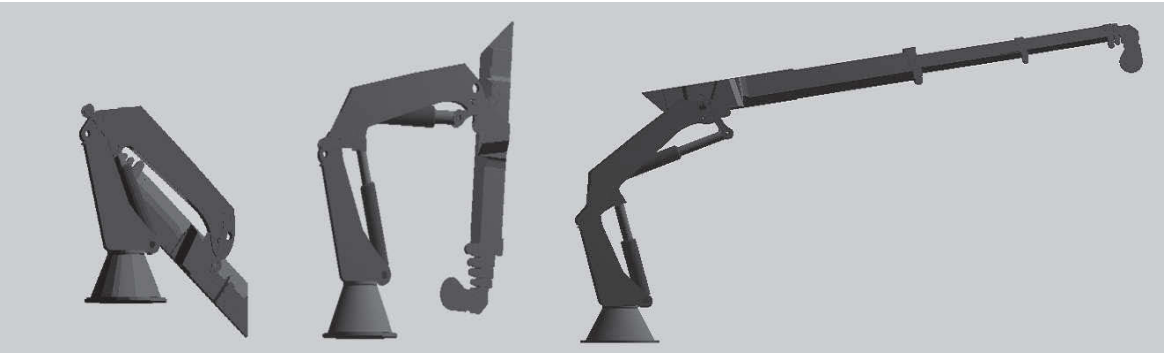


Figure 9: *Palfinger PK45000 Crane*

The two prevalent methods used to solve forward dynamics problems are the Articulated Body Algorithm (ABA) and the Composite Rigid Body Algorithm (CRBA) [9, 10, 11]. The ABA has a computational complexity of $O(N)$ (on the order of N operations to compute, where N is the number of links) while the CRBA has a computational complexity of $O(N^3)$. It has been shown that the ABA requires less computation than the CRBA for mechanisms with more than 3 DOF [12]. This makes the ABA the ideal method for the Palfinger PK 45000 boom crane that has 7 DOF.

In the next sections, the selection of the joint frame convention used is presented followed by a description of the the ABA and implementation.

2.4.2 Boom Crane Joint Frames

Given an articulated mechanism such as the one shown in Figure 10, there are an infinite number of ways to define frames of reference for each of the links in the mechanism. The Denavit-Hartenberg (D-H) convention provides a convenient and consistent methodology for providing a minimal representation with only four parameters per link. By applying the D-H convention, a simple table of parameters is all that is required to define the joint locations and degrees of freedom in the mechanism. Each row in the four column table specifies information about a link in the mechanism.

For the `SMS::Boomcrane` class, it was decided that the joint types and locations would be specified using the mechanism's unique table of defining D-H parameters. Although it is not necessary to use the D-H parameters to implement the forward dynamics solution of the ABA, the D-H parameters are a common convention used in expressing the state of a mechanism.

There are two D-H Parameter conventions: Craig's convention [13] and Paul's convention [14, 15]. For this work, it was determined that Paul's convention would be used. The reason for this choice will be discussed once the basics of the D-H Parameters and definition of the frames of reference are presented.

To determine the table of D-H parameters for a mechanism, a right handed orthogonal coordinate frame is attached at each joint, where the $\hat{\mathbf{z}}$ axis of the frame always points

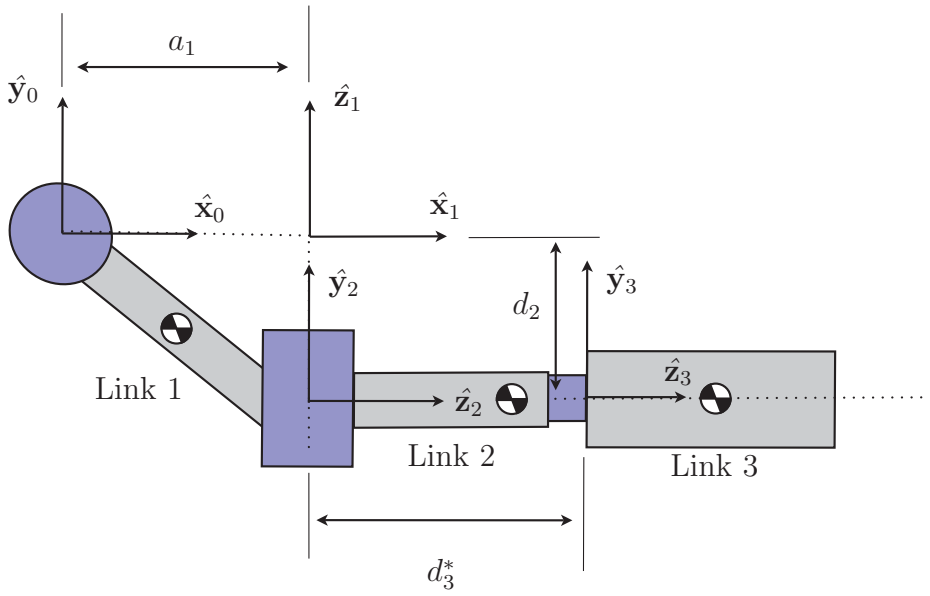


Figure 10: *The frames of an example boom crane whose D-H Parameters are found in Table 1 where the variable θ_2 is shown with deflection of* $90°$.

along the axis of actuation. The frames of reference are referred to as joint frames. The links closest to the base are referred to as proximal joints, and the links furthest from the base (closest to the end effector) are referred to as distal joints. Each consecutive joint frame i is attached to link i's distal joint. The joint frame for the base (frame 0) is positioned at the proximal joint of link 1.

D-H parameters define four convenient transformations that allow one to navigate between the frames of reference for each link.

- The first transformation is a translation, $\mathbf{A}_{d_i}^{(4\times4)}$, of frame $i-1$ along $\hat{\mathbf{z}}_{i-1}$ by an amount d_i until the center of the created intermediate transformed frame is located at the common normal between $\hat{\mathbf{z}}_{i-1}$ and $\hat{\mathbf{z}}_i$.

- The second is a rotation, $\mathbf{A}_{\theta_i}^{(4\times4)}$, of the intermediate transformed frame about the axis $\hat{\mathbf{z}}_{i-1}$ by an amount θ_i such that the $\hat{\mathbf{x}}$ axis of the newly transformed intermediate frame becomes co-linear with the common normal between the axes $\hat{\mathbf{z}}_{i-1}$ and $\hat{\mathbf{z}}_i$.

- The third is a translation, $\mathbf{A}_{a_i}^{(4\times4)}$, along the $\hat{\mathbf{x}}$ axis of the intermediate transformed frame by an amount a_i until the center of the newly created intermediate transformed frame becomes located at the center of link i's distal joint frame.

Link i	α_i	a_i	d_i	θ_i
1	-90	a_1	0	θ_1^*
2	90	0	d_2	θ_2^*
3	0	0	d_3^*	0

Table 1: *D-H Parameters and dynamics information for the sample boom crane in Figure 10.*

- The fourth is a rotation, $\mathbf{A}_{\alpha_i}^{(4\times4)}$, by an amount α_i about $\hat{\mathbf{x}}$ axis of the intermediate transformed frame until its $\hat{\mathbf{z}}$ axis points along $\hat{\mathbf{z}}_i$ of frame i.

Figure 10 shows a sample articulated body with three joints where each joint frame is defined using Paul's D-H parameter convention. The D-H Parameters for this particular articulated body are found in Table 1. The character "*" signifies the parameter for that link's joint that is the dynamic variable. The dynamic variable can only be the angle θ for revolute joints or distance d for prismatic joints.

Each row of Paul's D-H parameter table (see Table 1), holds the information about the 4 D-H parameter transformations that define each frame relative to the last. These transformations, the product of four individual transformations are defined as:

$$\mathbf{A}_{i-1\to i} = \mathbf{A}_{d_i}^{(4\times4)}\mathbf{A}_{\theta_i}^{(4\times4)}\mathbf{A}_{a_i}^{(4\times4)}\mathbf{A}_{\alpha_i}^{(4\times4)}, \tag{28}$$

$$\begin{aligned}\mathbf{A}_{i-1\to i} = &\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} c\theta_i & -s\theta_i & 0 & 0 \\ s\theta_i & c\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ &\times \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c\alpha_i & -s\alpha_i & 0 \\ 0 & s\alpha_i & c\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\end{aligned} \tag{29}$$

where $s\theta_i$ and $c\theta_i$ signify the sine and cosine of θ_i, respectively. Similarly, $s\alpha_i$ and $c\alpha_i$ signify the sine and cosine of α_i, respectively

The contents of the 4×4 matrix $\mathbf{A}_i$ consists of:

$$\mathbf{A}_{i-1\to i} = \begin{bmatrix} \mathbf{R}_{i\to i-1} & {}^{i-1}\mathbf{P}_{i-1\to i} \\ 0_{1\times3} & 1 \end{bmatrix} \tag{30}$$

where $\mathbf{R}_{i\to i-1}$ is a rotation matrix describing the orientation of reference frame i with respect to reference frame $i-1$, and consequently, it transforms the description of a vector about frame i to a description in frame $i-1$; the vector does not physically move,

and only the description changes as it is expressed with respect to a different reference frame. A rotation matrix has two uses: a passive rotation and an active rotation. A passive rotation signifies a change of frames of description, while an active rotation, does not alter the description frame but rather literally rotates a vector.

The vector ${}^{i-1}\mathbf{P}_{i-1\to i}$ is the position of the origin of frame i with respect to the origin of frame $i-1$, and described with respect to frame $i-1$. It is derived using the D-H parameters convention:

$$\mathbf{R}_{i\to i-1} = \mathbf{R}^T_{i-1\to i} = \begin{bmatrix} c\theta_i & -s\theta_i c\alpha_i & s\theta_i s\alpha_i \\ s\theta_i & c\theta_i c\alpha_i & -c\theta_i s\alpha_i \\ 0 & s\alpha_i & c\alpha_i \end{bmatrix} \tag{31}$$

and

$${}^{i-1}\mathbf{P}_{i-1\to i} = \begin{bmatrix} a_i c\theta_i \\ a_i s\theta_i \\ d_i \end{bmatrix} \tag{32}$$

Take note that Equation 30 has a negative ${}^{i-1}\mathbf{P}_{i\to i-1}$ because the Equation 29 produces a position translation vector, starting from frame $i-1$ to frame i rather than a vector that points from frame i to $i-1$.

2.4.2.1 Choice of Paul's D-H Parameter Convention

For a given link i, the standard (Paul's) convention fixes a reference frame to the distal joint of the link. In the modified (Craig's) convention, a reference frame is fixed to the proximal joint frame, as shown in Figure 11. For Paul's convention the transformation matrix $\mathbf{A}_{i\to i-1}$ defines the position and orientation of a link's fixed distal joint frame with respect to the distal joint frame of the previous link, which is also taken as the proximal joint frame of the current link. This makes use of the current link's length and twist parameters. For Craig's convention, the transformation matrix $\mathbf{A}_{i\to i-1}$ defines the position and orientation of a link's fixed proximal frame with respect to the previous links proximal joint frame, requiring the use of the previous link's length and twist parameters.

In terms of the `SMS::Boomcrane` implementation, each link in a mechanism is an `SMS::ABA_RigidBody` object having its own set of parameters (mass, inertia, etc) that define it. Thus to support the `SMS::ABA RigidBody`-centered approach in the computer code, Paul's convention was implemented because the required parameters (link length and twist) are naturally encapsulated with the current link.

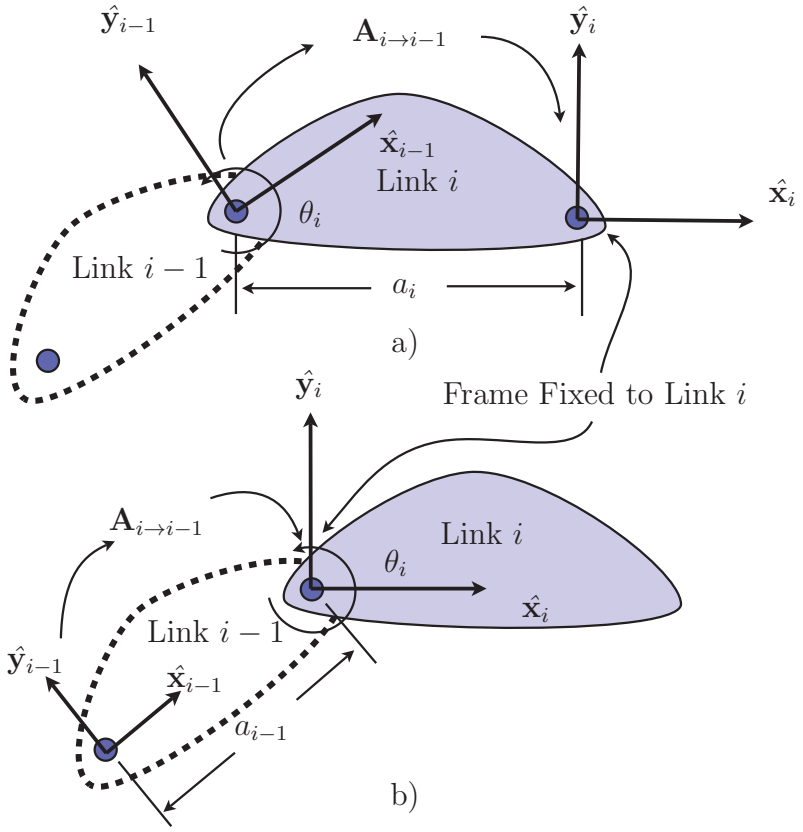


Figure 11: *Denavit-Hartenberg Conventions: a) Paul's convention, b) Craig's convention.*

2.4.3 Articulated Body Algorithm Overview

The ABA based Boom Crane model was implemented and encapsulated in the `SMS::BoomCrane` class, which evaluates the forward dynamics of the articulated body and determines the crane motion through time. The algorithm must allow for the addition of arbitrary disturbance forces to be applied to links so that the consequences of cable attachments and other effects can be simulated. The implementation of the ABA in the `SMS::BoomCrane` class is presented below. The implementation was based on the work presented in the literature [9, 11].

The state of an articulated body in the ABA is described in terms of the joint displacement for each link, which is θ_i for revolute joints or d_i for prismatic joints. The ABA advances the dynamics of the mechanism by calculating the joint accelerations and velocities, which are then integrated through time to produce future velocities and positions. To accomplish this, the dynamics of the articulated body is evaluated in three passes:

- The first pass starts at the first proximal joint frame and calculates the positions and velocities of each successive joint.
- Using the joint positions and velocities, the second pass starts at the last proximal joint and calculates the accumulated articulated inertias and forces seen at each joint.
- The third and final pass starts at the first proximal joint and calculates the successive resulting joint accelerations based on the known inertias and forces determined previously.

The following section describes how the velocities, inertial properties, forces, and accelerations of the links are determined recursively and used to evaluate the dynamics of a mechanism.

2.4.4 Joint Velocities

To complete the first pass of the ABA, referred to as forward kinematics, the velocities of each link's joint must be calculated. The first pass is also used to calculate velocity dependent terms, $\zeta_p^{(i)}$ and $\beta_p^{(i)}$, which are discussed in section 2.4.5 and 2.4.6.

Consider Figure 12, which defines the joint frames of link i using D-H Parameters. Frame p is link i's proximal joint frame and frame d is link i's distal joint frame. If frame p of link i is taken to coincide exactly with frame d of link $i-1$ for all time, then link i's distal joint d frame can be related to the proximal joint frame p using the D-H Parameters based rotation matrix, $\mathbf{R}_{d\to p}^{(i)}$ (akin to $\mathbf{R}_{i\to i-1}$):

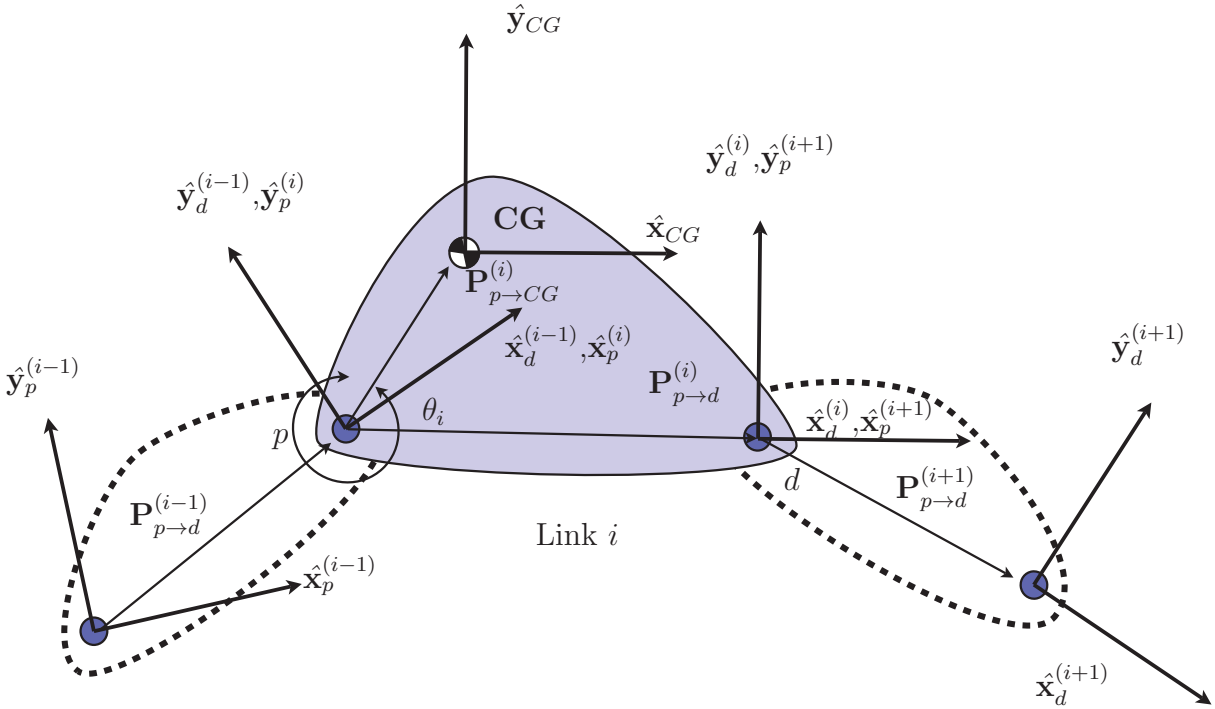


Figure 12: *The frames of an articulated body link. Link $i-1$ and $i+1$ shadowed here for visualization purposes.*

$$\mathbf{R}_{d\to p}^{(i)} = \left(\mathbf{R}_{p\to d}^{(i)}\right)^T = \begin{bmatrix} c\theta_i & -s\theta_i c\alpha_i & s\theta_i s\alpha_i \\ s\theta_i & c\theta_i c\alpha_i & -c\theta_i s\alpha_i \\ 0 & s\alpha_i & c\alpha_i \end{bmatrix} \tag{33}$$

The position vector ${}^i\mathbf{P}_{p\to d}$ (akin to ${}^i\mathbf{P}_{i-1\to i}$), which is the distal joint frame with respect to the proximal joint frame, is:

$${}^i\mathbf{P}_{p\to d} = \begin{bmatrix} a_i c\theta_i \\ a_i s\theta_i \\ d_1 \end{bmatrix} \tag{34}$$

The velocity of a link is expressed in a six-dimensional vector space as a 6×1 array that holds the three Cartesian linear velocity components and three Cartesian angular velocity components. The velocity of link i's proximal joint, joint p, can be defined as the sum of the velocity components from the previous links distal joint, plus the velocity component provided by the joint itself:

$$\mathbf{v}_p^{(i)} = \mathbf{v}_d^{(i-1)} + \phi_p^{(i)} \dot{\varepsilon}_p^{(i)} \tag{35}$$

Here, $\phi_p^{(i)} = [0,0,1,0,0,0]^T$ for a prismatic joint, $\phi_p^{(i)} = [0,0,0,0,0,1]^T$ for a revolute joint, and $\varepsilon_p^{(i)}$ and $\dot{\varepsilon}_p^{(i)}$ are joint i's displacement and rate of displacement, respectively. The joint displacement, ε, has units of either radians or meters for revolute or prismatic joints, respectively. In the case of the first link, $\mathbf{v}_d^{(i-1)}$ is the velocity of the base.

The velocity of link i's distal joint frame, $\mathbf{v}_d^{(i)}$, can be determined given $\mathbf{v}_p^{(i)}$:

$$\mathbf{v}_d^{(i)} = \mathbf{T}_{p\to d}^{(v)(i)} \ \mathbf{v}_p^{(i)} \tag{36}$$

where $\mathbf{T}_{p\to d}^{(v)(i)}$ is a 6×6 transformation matrix that provides the linear and rotational velocity of joint d by considering the orientation and translation difference of the frames p and d:

$$\mathbf{T}_{p\to d}^{(v)(i)} = \begin{bmatrix} \mathbf{R}_{p\to d}^{(i)} & \mathbf{R}_{p\to d}^{(i)} (\tilde{\mathbf{P}}_{p\to d}^{(i)})^T \\ 0 & \mathbf{R}_{p\to d}^{(i)} \end{bmatrix} \tag{37}$$

where $\tilde{\mathbf{P}}_{p\to d}^{(i)}$ is the skew-symmetric form of the vector $\mathbf{P}_{p\to d}^{(i)}$. The skew-symmetric form of a vector is used to evaluate the cross-product of the vector with another vector using vector-matrix multiplication. The multiplication $\tilde{\mathbf{p}}\mathbf{v}$ produces the same results as the cross product operation $\mathbf{p} \times \mathbf{v}$ while either $\tilde{\mathbf{p}}^T\omega$ or $-\tilde{\mathbf{p}}\omega$ signifies the cross product operation $\omega \times \mathbf{p}$. Here, the 3×3 $\mathbf{R}_{p\to d}^{(i)} (\tilde{\mathbf{P}}_{p\to d}^{(i)})^T$ in the upper right quadrant of the velocity transformation matrix operates on the angular velocity, which provides the linear velocity component of joint d that arises from the rotation of link i with respect to the proximal frame. The velocity at the distal joint, $\mathbf{v}_d^{(i)}$, is provided to the next link for its velocity calculations.

2.4.5 Articulated Forces and Inertias

The second pass of the ABA, referred to as backward dynamics, calculates the bias forces $\beta_p^{(i)}$ and articulated inertias $\mathbf{M}_p^{*(i)}$ about each link's proximal joint. The inertias and forces seen at each proximal joint are dependent on the inertia and forces of the link plus those propagated through the distal joints. Both the forces and inertia experienced by a link's proximal joint are required to evaluate the joint accelerations in the third and final pass detailed in Section 2.4.6.

To develop the equations that will provide the required information, link i's rigid body frame, which is fixed at the CG of the link, and with respect to which the link's inertial information is defined, is first defined relative to its proximal frame. On link i the rotation matrix $\mathbf{R}^{(i)}_{p\to CG}$ defines the orientation of the rigid body frame relative to frame p:

$$\mathbf{R}^{(i)}_{p\to CG} = \begin{bmatrix} c\psi c\theta & c\psi s\theta s\phi - s\psi c\phi & s\psi s\phi + c\psi s\theta s\phi \\ s\psi c\theta & s\psi s\theta s\phi + c\psi c\phi & s\theta s\psi c\phi - c\psi s\phi \\ -s\theta & c\theta s\phi & c\theta c\phi \end{bmatrix} \tag{38}$$

where ψ, θ, and ϕ are the $\hat{\mathbf{z}}_i$, $\hat{\mathbf{y}}'_i$, $\hat{\mathbf{x}}''_i$ sequence Euler angles that describe orientation. The location and orientation of the CG relative to frame p is defined as:

$${}^{i}\mathbf{P}_{p\to CG} = \begin{bmatrix} x_{p\to CG} & y_{p\to CG} & z_{p\to CG} \end{bmatrix}^T \tag{39}$$

The displacements of the joints will cause motion of any outboard, or downstream, links but will have no effect on that particular proximal joint's frame because of how the D-H Parameters have been defined (that is, because the proximal joint frame is in effect fixed to the distal joint of the previous link). To obtain the CG location and orientation of the rigid body relative to frame p, a joint actuation transformation must be applied. For revolute joints, the actuated relative position and orientation, $\mathbf{R}'^{(i)}_{p\to CG}$ and $\mathbf{P}'^{(i)}_{p\to CG}$, are:

$$\mathbf{R}'^{(i)}_{p\to CG} = \mathbf{R}^{(i)}_{\theta}\mathbf{R}^{(i)}_{p\to CG} \tag{40}$$

and

$$\mathbf{P}'^{(i)}_{p\to CG} = \mathbf{R}^{(i)}_{\theta}{}^{i}\mathbf{P}_{p\to CG} \tag{41}$$

where

$$\mathbf{R}^{(i)}_{\theta} = \begin{bmatrix} c\varepsilon^{(i)}_p & s\varepsilon^{(i)}_p & 0 \\ -s\varepsilon^{(i)}_p & c\varepsilon^{(i)}_p & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{42}$$

For prismatic joints, the actuated relative position and orientation, $\mathbf{R}'^{(i)}_{p\to CG}$ and $\mathbf{P}'^{(i)}_{p\to CG}$, are:

$$\mathbf{R}'^{(i)}_{p\to CG} = \mathbf{R}^{(i)}_{p\to CG} \tag{43}$$

and

$$\mathbf{P}'^{(i)}_{p\to CG} = {}^{i}\mathbf{P}_{p\to CG} + {}^{i}\mathbf{P}_{displ} \tag{44}$$

where ${}^{i}\mathbf{P}_{displ} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \varepsilon_p^{(i)}$.

The 6×6 mass and moment of inertia tensor $\mathbf{M}^{(i)}_{CG}$ for the link's rigid body is described about the rigid body frame as:

$$\mathbf{M}^{(i)}_{CG} = \begin{bmatrix} \mathbf{m}^{(i)} & 0_{3\times 3} \\ 0_{3\times 3} & \mathbf{I}^{(i)}_{CG} \end{bmatrix} \tag{45}$$

where $\mathbf{I}^{(i)}_{CG}$ is the moment of inertia tensor about the rigid body's frame for link i, and $\mathbf{m}^{(i)}$ is the 3×3 rigid body mass tensor.

The mass moment of inertia of link i about frame p is defined making use of the parallel axis theorem [11]:

$$\mathbf{I}^{(i)}_{p} = \mathbf{R}'^{(i)}_{CG\to p}\mathbf{I}^{(i)}_{CG}\mathbf{R}'^{(i)}_{p\to CG} + m^{(i)}(\mathbf{P}'^{(i)}_{p\to CG}\cdot\mathbf{P}'^{(i)}_{p\to CG}\hat{\mathbf{I}} - \mathbf{P}'^{(i)}_{p\to CG}\otimes\mathbf{P}'^{(i)}_{p\to CG}) \tag{46}$$

where $m^{(i)}$ is the rigid body mass, $\hat{\mathbf{I}}$ is a 3×3 identity matrix, and the outer product of two vectors is described using the $\otimes$ operator. The outer product operation is defined as $\mathbf{u}\otimes\mathbf{w} = \mathbf{u}\mathbf{w}^T$.

The dynamics of a link are evaluated at the proximal joint frame. Therefore, the required 6×6 mass and moment of inertia tensor $\mathbf{M}^{(i)}_{p}$ for the rigid body described about frame p is defined as:

$$\mathbf{M}^{(i)}_{p} = \begin{bmatrix} \mathbf{m}^{(i)} & \tilde{\mathbf{h}}^{(i)T} \\ \tilde{\mathbf{h}}^{(i)} & \mathbf{I}^{(i)}_{p} \end{bmatrix} \tag{47}$$

where $\tilde{\mathbf{h}}^{(i)}$ is the 3×3 skew symmetric form of the vector $\mathbf{h}^{(i)} = \mathbf{m}^{(i)}\mathbf{P}'^{(i)}_{p\to CG}$, which accounts for linear and angular inertial contributions due to the link CG location distance from the proximal joint, and $\mathbf{I}^{(i)}_{p}$ is the mass moment of inertia of link i about joint p as described in Equation 46.

The acceleration observed at link i's joint p caused by the articulated inertia, bias forces, and any external forces is described by the following equation:

$$\mathbf{f}^{(i)}_{p} = \mathbf{M}^{*(i)}_{p}\mathbf{a}^{(i)}_{p} - \beta^{*(i)}_{p} \tag{48}$$

where $\mathbf{M}_p^{*(i)}$ is called the articulated inertia [9], $\beta_p^{*(i)}$ is called accumulated bias force, and $\mathbf{f}_p^{(i)}$ is an externally applied force. Bias forces are best described as the forces that must be applied to a link to maintain particular link velocities.

The articulated inertia $\mathbf{M}_p^{*(i)}$, see Figure 13, is the accumulated inertia of the mechanism downstream of link i, including link i's own inertia $\mathbf{M}_p^{(i)}$, expressed about that link's proximal joint frame:

$$\mathbf{M}_p^{*(i)} = \mathbf{M}_p^{(i)} + (\mathbf{T}_{p\to d}^{(f)(i)})^T \mathbf{N}_d^{(i)} \mathbf{T}_{p\to d}^{(f)(i)} \tag{49}$$

where $\mathbf{N}_d^{(i)} = \mathbf{N}_p^{(i+1)}$ and is the articulated inertia seen at the joint due to all downstream links with the inertia about each downstream joint DOF removed:

$$\mathbf{N}_p^{(i)} = \mathbf{M}_p^{*(i)} + \mathbf{k}_p^{(i)} \otimes (\frac{-1}{m_p^{*(i)}} \mathbf{k}_p^{(i)}) \tag{50}$$

$$\mathbf{k}_p^{(i)} = \mathbf{M}_p^{*(i)} \phi_p^{(i)} \tag{51}$$

and

$$m_p^{*(i)} = \left(\phi_p^{(i)}\right)^T \mathbf{M}_p^{*(i)} \phi_p^{(i)} \tag{52}$$

Note that $m_p^{*(i)}$ is the joint i inertia and governs the resulting joint acceleration; it has units of rotational inertia for revolute joints and units of mass for prismatic joints. Equations 49 through 52 highlight how the joint inertia is a function of the orientation and position of the links between joint i and the end effector.

To propagate force contributions from joint d on to joint p, a similar transformation matrix to the velocity transform $\mathbf{T}_{p\to d}^{(v)}$ called $\mathbf{T}_{d\to p}^{(f)}$ transforms forces from joint d to joint p and accounts for differences in frame orientations as well as moments generated by linear forces and the distance between the frames:

$$\mathbf{T}_{d\to p}^{(f)(i)} = \left(\mathbf{T}_{p\to d}^{(v)(i)}\right)^T = \begin{bmatrix} \mathbf{R}_{p\to d}^{(i)} & 0 \\ \tilde{\mathbf{P}}_{p\to d}^{(i)} \mathbf{R}_{p\to d}^{(i)} & \mathbf{R}_{p\to d}^{(i)} \end{bmatrix} \tag{53}$$

The accumulated bias force $\beta_p^{*(i)}$ is defined recursively as follows:

$$\beta_p^{*(i)} = \beta_p^{(i)} + \mathbf{T}_{d\to p}^{(f)(i)} \beta_d^{(i)} \tag{54}$$

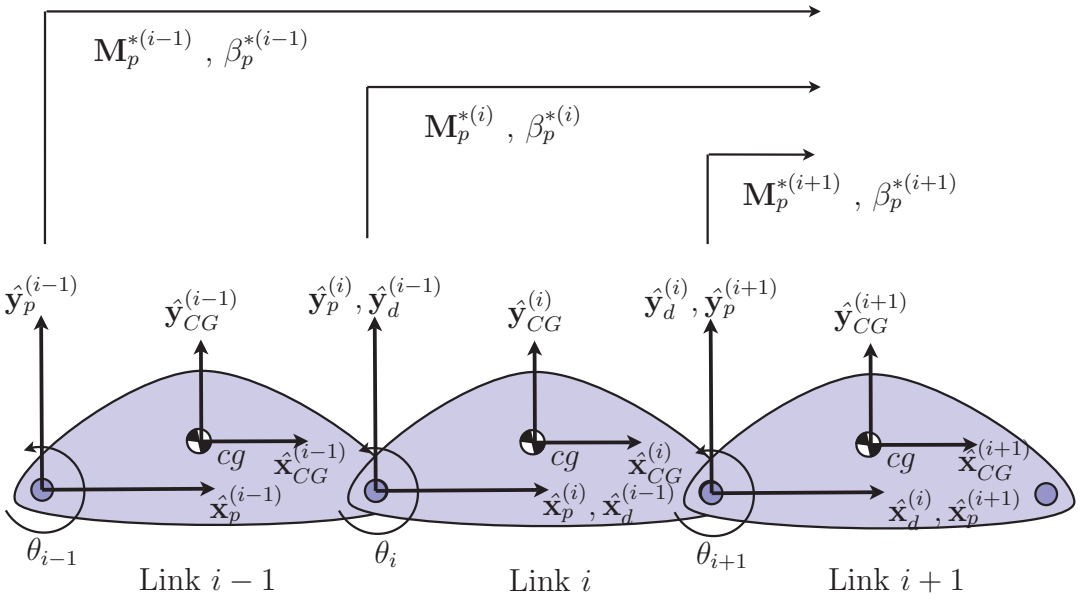


Figure 13: *The accumulation of mass and bias forces for an articulated body.*

where $\beta_p^{(i)}$ is the bias force contribution from this rigid body link:

$$\beta_p^{(i)} = \begin{bmatrix} \omega_p^{(i)} \times (\omega_p^{(i)} \times \mathbf{h}^{(i)}) \\ \omega_p^{(i)} \times \mathbf{I}_p^{(i)} \omega_p^{(i)} \end{bmatrix} \tag{55}$$

and

$$\beta_d^{(i)} = \beta_p^{*(i+1)} - \mathbf{N}_p^{(i+1)} \zeta_p^{(i+1)} - \mathbf{k}_p^{(i+1)} \frac{1}{m_p^{*(i+1)}} \tau_p^{*(i+1)} \tag{56}$$

Note that $\tau_p^{*(i)}$ is the total force, including joint actuation force, in the actuation direction of joint i and is necessary to determine the joint acceleration. The variable $\zeta_p^{(i)}$ is a velocity and joint-type dependent acceleration term and is discussed in the following section. Equation 56 indicates how the equal and opposite reaction force resulting from joint forcing and appropriate acceleration effects are applied on proximal rigid bodies.

2.4.6 Joint Acceleration

The third and last pass, called the forward acceleration pass, consists of calculating the accelerations of each joint of the mechanism $\mathbf{a}_p^{(i)}$ and $\ddot{\varepsilon}_p^{(i)}$ given all of the forces

accumulated at the joint and the articulated inertias that influence the resulting joint acceleration.

The velocity dependent acceleration term $\zeta_p^{(i)}$, which is calculated during forward kinematics, is defined as:

$$\zeta_p^{(i)} = \zeta_d^{(i-1)} + \zeta_\varepsilon^{(i)} \tag{57}$$

where $\zeta_\varepsilon^{(i)}$ is the acceleration contribution resulting from particular joint type velocities. For revolute joints:

$$\zeta_\varepsilon^{(i)} = \begin{bmatrix} 0 \\ \omega_p^{(i)} \times \dot{\varepsilon}_i \phi_p^{(i)} \end{bmatrix} \tag{58}$$

and for prismatic joints:

$$\zeta_\varepsilon^{(i)} = \begin{bmatrix} 2(\omega_p^{(i)} \times \dot{\varepsilon}_i \phi_p^{(i)}) \\ 0 \end{bmatrix} \tag{59}$$

There is also a component of the acceleration, $\zeta_d^{(i)}$, that is dependent on the velocity of the distal joint of the upstream link:

$$\zeta_d^{(i)} = \begin{bmatrix} \mathbf{R}_{p\to d}^{(i)} \omega_p^{(i)} \times (\omega_p^{(i)} \times^i \mathbf{P}_{p\to d}) \\ 0 \end{bmatrix} \tag{60}$$

The total joint force, $\tau_p^{*(i)}$ is:

$$\tau_p^{*(i)} = \tau_p^{(i)} + \left(\phi_p^{(i)}\right)^T \beta_p^{*(i)} \tag{61}$$

where:

$$\tau_p^{(i)} = \left(\phi_p^{(i)}\right)^T \mathbf{f}_p^{(i)} \tag{62}$$

Forces, such as those from an attached cable, can be applied to any of the boom crane rigid body links. The force is transformed using $\mathbf{T}_{CG\to p}^{(f)(i)}$, defined similarly to $\mathbf{T}_{d\to p}^{(f)(i)}$, to obtain the resulting force applied about frame p, $\mathbf{f}_p^{(i)}$. A user-defined force designed to

actuate a joint by a controller or elastic end stop can also be applied in a similar fashion by directly superimposing it on $\tau_p^{(i)}$, which is the force in the actuation direction of the joint. Note that when link i has no downstream link, $\mathbf{M}_{d\to p}^{(i)}$ and $\beta_{d\to p}^{(i)}$ are zero matrices.

Recalling equation 48, the acceleration for joint p, $\mathbf{a}_p^{(i)}$, is quantified:

$$\mathbf{a}_p^{(i)} = \mathbf{a}_d^{(i-1)} + \phi_p^{(i)}\ddot{\varepsilon}_i + \zeta_p^{(i)} \tag{63}$$

where $\mathbf{a}_d^{(i)}$ is the acceleration of link (i)'s distal joint obtained in an analogous manner to the velocities:

$$\mathbf{a}_d^{(i)} = \mathbf{T}_{p\to d}^{(v)(i)}\mathbf{a}_p^{(i)} \tag{64}$$

and $\ddot{\varepsilon}_i$ is the joint acceleration, defined as:

$$\ddot{\varepsilon}_i = (m_p^{*(i)})^{-1}\left[\tau_i^{*(i)} - \left(\phi_p^{(i)}\right)^T \mathbf{M}_p^{*(i)}(\mathbf{a}_d^{(i-1)} + \zeta_p^{(i)})\right] \tag{65}$$

For the case of link 1, the proximal neighbouring link's acceleration contributions are obtained from the boom crane base acceleration: $\mathbf{a}_d^{(i-1)} = \mathbf{a}_{base}$.

To summarize, the ABA calculates the accelerations of the joints of an articulated body in three passes. The first pass, forward kinematics, starts at the base frame and calculates the cumulative positions and velocities of each successive joint $\mathbf{v}_p^{(i)}$ and velocity dependent terms such as $\beta_p^{(i)}$ and $\zeta_p^{(i)}$. The second pass, backward dynamics, starts at the last link and works backward to the first link while quantifying all of the accumulated articulated inertias $\mathbf{M}_p^{*(i)}$ and forces $\beta_p^{*(i)}$. The third and final pass, forward accelerations, starts at the first link and calculates the joint degree of freedom accelerations, $\ddot{\varepsilon}_p^{(i)}$, which are then numerically integrated through time to propagate the state of the crane. The forward acceleration pass also quantifies the successive absolute accelerations seen at the joint, $\mathbf{a}_p$, which are needed to evaluate downstream joint degree of freedom accelerations.

2.4.7 Base Frame

Note that the position and orientation of the base frame was not defined relative to any other frame using the D-H parameters; to properly define the serial articulated body chain with respect to other objects, the orientation of the base frame relative to the Earth-fixed frame is defined using Euler angles:

$$\mathbf{R}_{p\to E}^{(1)} = \begin{bmatrix} c\psi c\theta & c\psi s\theta s\phi - s\psi c\phi & s\psi s\phi + c\psi s\theta s\phi \\ s\psi c\theta & s\psi s\theta s\phi + c\psi c\phi & s\theta s\psi c\phi - c\psi s\phi \\ -s\theta & c\theta s\phi & c\theta c\phi \end{bmatrix} \tag{66}$$

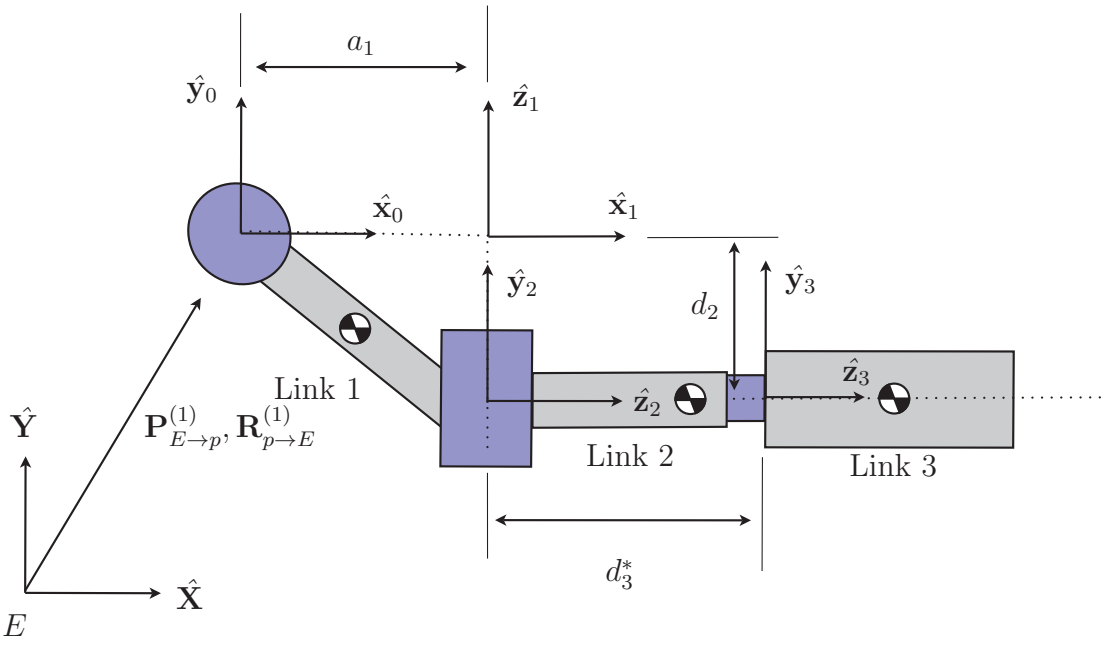


Figure 14: *The location and orientation of the boom crane base frame relative to the Earth-fixed frame.*

where ψ, θ, and ϕ are Euler angles rotated about the non-orthogonal axes $\hat{\mathbf{z}}_i$, $\hat{\mathbf{y}}'_i$, $\hat{\mathbf{x}}''_i$ respectively. The base frame position is defined using the Cartesian vector $\mathbf{P}^{(1)}_{E\to p}$ (see Figure 14).

2.4.8 PID Joint Control

A joint PID controller can be added to any joint that will superimpose forces to $\tau_p^{(i)}$ to control that joint. The controller can apply forces in an effort to control the joint's position or velocity.

For the position controller, the force applied to a joint to maintain a target position is:

$$\tau^{(i)}_{PID} = C^{(i)}_p(\varepsilon^{(i)}_{target} - \varepsilon^{(i)}_p) + C^{(i)}_I(\sum(\varepsilon^{(i)}_{target} - \varepsilon^{(i)}_p)dt) - C^{(i)}_d(\dot{\varepsilon}^{(i)}_p) \tag{67}$$

where $C^{(i)}_p$ is the proportional coefficient, $C^{(i)}_I$ is the integral coefficient, $C^{(i)}_d$ is the damping coefficient for joint i, $\varepsilon^{(i)}_{target}$ is the joint position set point for joint i, and dt is the simulation integrator time step.

For a velocity controller, the force added to $\tau_p^{(i)}$ becomes:

$$\tau_{PID}^{(i)} = C_p^{(i)}(\dot{\varepsilon}_{target}^{(i)} - \dot{\varepsilon}_p^{(i)}) + C_I^{(i)}(\sum(\dot{\varepsilon}_{target}^{(i)} - \dot{\varepsilon}_p^{(i)})dt) - C_d^{(i)}(\ddot{\varepsilon}_p^{(i)}) \tag{68}$$

The force added to the joint by the PID controller is limited by a maximum power that controller can use to actuate a joint. The power, $P^{(i)}$, used to actuate a joint is:

$$P^{(i)} = \tau_{PID}^{(i)}\dot{\varepsilon}_p^{(i)} \tag{69}$$

hence the maximum allowable controller force becomes:

$$\tau_{PID,max}^{(i)} = \frac{P_{max}^{(i)}}{\dot{\varepsilon}_p^{(i)}} \tag{70}$$

2.4.9 Limiting Joint Range

Joint limits can be enforced by adding a force to a joint's $\tau_p^{(i)}$. The force added to $\tau_p^{(i)}$ to enforce a violated lower limit is:

$$\tau_{LL}^{(i)} = -K^{(i)}(\varepsilon_p^{(i)} - \varepsilon_{LL}^{(i)}) - D^{(i)}(\dot{\varepsilon}_p^{(i)}) \tag{71}$$

and for an upper limit:

$$\tau_{UL}^{(i)} = K^{(i)}(\varepsilon_p^{(i)} - \varepsilon_{UL}^{(i)}) - D^{(i)}(\dot{\varepsilon}_p^{(i)}) \tag{72}$$

where $\varepsilon_{UL}^{(i)}$ and $\varepsilon_{LL}^{(i)}$ are the joint's i's upper and lower actuation limits, and $K^{(i)}$ and $D^{(i)}$ are the joint limit stiffness and damping coefficients for joint i respectively.

2.5 Contact Resolution

During the course of a SMS operation simulation, an `SMS::Payload`, possibly supsended from an `SMS::Boomcrane`'s `SMS::Cable`, has the potential to come into contact with the ship's deck, hull, or transom. Likewise, in the simulation of shipborne operations involving a payload, winch, winch cable, and a crane, there is potential for the payload to collide with other objects in the simulation. The following section presents the strategy adopted within the SMS API to detect collisions and resolve the contact forces between the payload and the ship within a simulation.

The efficient and accurate simulation of such contacts is a field of study that encompasses many disciplines [16]. In general, resolving contacts in such multi-body simulations consists of two main problems that must be solved:

- detecting collisions between bodies
- determining the resulting contact forces

Many different solutions have been proposed to solve these problems [2, 16, 17, 18, 19]. Finding the optimal solution to these problems generally consists of finding a balance between accuracy and efficiency based on the needs of the application. In the following sections, an introduction to collision detection and contact force resolution is provided. The theory behind the collision detection methods and contact models selected and implemented are then discussed.

2.5.1 Overview of Collision Detection

Bodies are considered to be in contact if they are interfering with each other or are sharing the same space. The purpose of any collision detection method is to determine:

- **if** bodies are in contact, *and/or*
- **when** bodies are in contact, *and/or*
- **where** bodies are in contact.

There are a variety of methods available that solve these three types of problems. Due to the diverse nature of contact problems, methods that are well-suited for one type of application may not be well-suited for another.

To develop a collision detection algorithm, it is necessary to first describe the geometry of the bodies of interest in some way. The geometrical representations of these objects have possibly the biggest impact on the choice of methods to use. Most methods have been developed and are available for polygonal representations, which is likely due to the the widespread use of graphics hardware that use triangles as rendering primitives [2]. By using polygonal representation of geometry in the SMS API, future work can leverage 3D graphical file formats such as *.3ds or *.vrml to construct contact objects.

Objects can be classified as either convex or concave. Convex objects are especially suited to collision detection because of some of their inherent mathematical properties [2]. Simulation of concave objects are discussed less in literature, which is likely due to the fact that concave objects can be effectively handled by decomposing them

into a set of convex objects. In the SMS API, it was decided to initially limit simulations to simple convex contact objects. In the future, it would be possible to simulate concave objects by decomposing the concave object into a series of concatenated convex objects.

For the SMS API, it was also decided to use a collision detection method that determines not only if collisions occur, but where collisions occur as well. This information is needed to resolve the collision forces accurately. The most common methods that determine where collisions occur are Minimum Separation Distance (MSD) algorithms. MSD algorithms are designed to determine whether or not a collision exists based on when the minimum separation distance between a pair of colliding objects is zero. Conveniently, MSD algorithms have been used to efficiently find collisions between both concave and convex objects [20, 21]. MSD algorithms also provide the benefit of knowledge of the closest geometric features (vertices, edges, or polygons), which is useful for accurately quantifying contact forces. The minimum separation distance can be described, as shown in Figure 15 a), as the distance between a pair of points, one in or on each object whose Cartesian separation distance is minimum, or equal to the minimum, of any other possible combination of points in or on their respective objects. Noteworthy MSD algorithms that were considered were:

- the V-Clip algorithm [2, 22],
- optimisation-based methods [2, 17],
- and the Gilbert-Johnson-Keerthi Algorithm (GJK) [2, 23].

Of these available choices, The GJK algorithm, which does not require geometries be stored in any specific format, was chosen for the SMS API because it is one of the most efficient methods for determining the intersection between two convex polyhedrons [2]. In addition to providing the MSD, the GJK algorithm robustly provides a useful common contact property called the penetration depth or Minimum Translational Distance (MTD). As shown in Figure 15 b), when two objects are colliding, the MTD is the minimum distance that two objects must be translated relative to each other to achieve object separation [16, 24]. Since the MTD provides information critical to determining the contact force, the GJK algorithm was an ideal collision detection candidate for the SMS API.

The complexity of the simulation environment must be taken into consideration. An example of a complex environment would be simulation with a large number of objects that could potentially collide. In the worst case, to ensure that all collisions are detected, each object should be checked against every other object in the simulation environment; this provides a worst case scenario with $O(N^2)$ pairwise collision checks for an N body simulation. To reduce some of this computational burden, spatial approximations (i.e.,

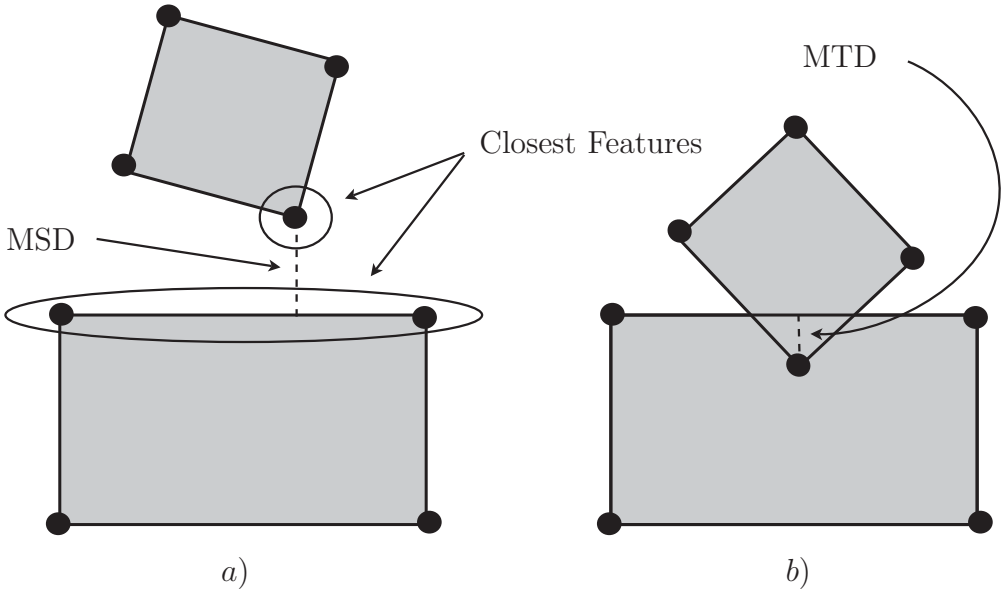


Figure 15: *a) The Minimum Separation Distance (MSD) between two objects showing the closest features are a vertex and a face, b) The Penetration Depth or Minimum Translational Distance.*

bounding volumes), partitioning schemes (i.e., bounding volume hierarchies, octrees, BSP-trees) and temporal strategies [2, 16, 17] are usually employed with success in real-time applications [2]. This is generally called broad-phase collision detection, where collisions you know are not occurring are pruned or culled from the set of possibly colliding objects, while narrow-phase collision detection is the act of determining if two objects are overlapping, such as by using the MSD. For the SMS API, only interactions between two objects are considered (i.e., the payload and an external object, such as the ship's deck or hull). This assumption limits the contact detection computational requirements without the need for any spatial approximations, etc. In the future, should a more complex simulation environment be constructed, these methods should be considered to alleviate the potential computational burden.

2.5.2 Overview of Contact Models

When a collision is occurring, contact forces must be determined and applied to resolve the dynamics of the bodies that are in contact. The contact force is typically resolved into its normal and tangential (or frictional) components.

2.5.2.1 Normal Contact Force Models

A variety of normal contact models have been proposed and they can generally be categorised as either discrete or continuous models [18, 25]. The most well-known discrete contact models are arguably the Newtonian impulse-momentum models or coefficient of restitution models [18]. Discrete models cannot be easily extended to multi-body contacts; moreover, the inclusion of friction for these models might violate energy conservation principles [25].

Continuous/compliant models integrate the contact forces over time. This allows contacting objects to deform under pressure as is the case for real contact situations. Continuous contact models can be categorized as those based on geometrical interference where the geometries do not actually deform (such as penetration depth models) and those that enforce boundary separation by modelling the geometrical deformations of the objects in contact (finite element methods). For some computer simulation applications, modelling pressure-based object deformations may not be practical due to the high associated computational cost, such as with finite element methods[26]. Fortunately, it can generally be assumed that the interfering geometry is a good approximation for the actual deformations involved given certain assumptions [18, 25, 27, 28, 29, 30].

Notable penetration depth models are the Hertzian contact model [31], the Kelvin-Voigt model [18, 32, 33, 34], the Hunt-Crossley model [29, 34], the Winkler elastic foundation model [31], the Gonthier model [26], and the Kelvin-Voigt-Winkler model [35].

The volume of interference models, such as the Gonthier model and the Kelvin-Voigt-Winkler model, make a link between the deformations of the Winkler elastic spring bed deformation and the volume of interference. Volume of interference contact force models are an advancement over the other penetration depth models because they free the user from having to worry about the shape of the contacting surfaces as that information is included in the interference geometry. As an example, the Hunt-Crossley model has geometry-dependent exponents, and the user needs to determine what those exponents are for every possible combination of contacting geometries before simulations can be completed.

To determine the normal contact force in the SMS API, the Kelvin-Voigt-Winkler model was selected because the damping component of the solution is based on internal material damping and deformation rates; the Gonthier damping component is dependant on knowledge of the coefficient of restitution a-priori and the Gonthier damping factor will vary with contact interface geometry, object materials, and impact velocity. Obtaining coefficients of resitution from experiment for varying contact geometries, material, and impact velocities can be impractical or even impossible. Contact events simulated

with the Kelvin-Voigt-Winkler model exhibit expected contact behaviour similar to Gonthier's model [35]. Examples of these contact behaviors, the coefficient of restitution as a function of impact velocity and contact force as a function of penetration depth are given in section 6.2 of [36]. The Kelvin-Voigt-Winkler model material properties are independent of the coefficient of restitution and should be constant regardless of contact geometry or impact velocity; the coefficient of restitution is dependent on the material properties only.

2.5.2.2 Tangential Contact Force Models

If two bodies are in contact and their surfaces are moving relative to each other at the contact location, friction forces can be present. A friction force model is required to determine the magnitude of these forces. Friction models are also categorizable into two types: static and dynamic. Static friction models only consider the current state of friction, while dynamic friction models include information from previous states.

The most notable static friction models are arguably the classical friction models. These are based around the Coulomb friction model, which states that for two objects whose contact surfaces are sliding relative to each other, the friction force magnitude is equal to the normal contact force multiplied by the Coulomb friction coefficient and applied in the direction opposing the relative velocity of the contacting objects. The classical models extend this to handle sticking friction as well viscous friction [19].

The Karnopp model was implemented in the SMS API. The Karnopp model is a classical model that uses a small velocity dead-zone to handle stick/slip transitions [19].

One dynamic friction model that is worth mentioning is the LuGre friction model [19, 37]. It models lubricated surfaces, the Stribeck effect, time-dependent break-away force, and frictional lag. This model was extended to handle three dimensional vectorial frictional forces and later spinning friction [26, 29]. Should a more sophisticated friction model be required, the LuGre friction model should be considered.

2.5.3 Summary of the SMS API Contact Dynamics Methods

The SMS API uses convex polyhedral geometric representations of the payload and an external object for contact resolution. The GJK algorithm was employed to detect collisions and provide MSD and MTD information. When collisions are detected, the Kelvin-Voigt-Winkler contact model is used to quantify the normal contact force and the Karnopp friction model is used to provide tangential friction forces though other models could be implemented as replacements. There are a few limitations to these particular choices:

- the local strains on the objects must remain small because of the engineering strain formulation of the normal contact force model,
- the internal stiffness of the object's materials are assumed linear with deflection,
- the material damping properties of the objects are assumed linear with deflection rate,
- the contact stresses are one dimensional and Poisson's ratio is ignored,
- rolling resistance and spinning friction are ignored, as are most frictional phenomena captured by more sophisticated friction models,
- the location or state of contact surface is ignored,
- most of the methods described here take advantage of special properties of convex polyhedral objects and are therefore not appropriate for concave objects.

The following section describes in some details the methods and models that were introduced in the overview and are used by the SMS API to resolve contact effects. First, Section 2.5.4 describes a point contact model framework that shows how contact forces are applied to rigid bodies, their effects on the dynamics of the rigid bodies, as well as how pertinent contact information, such as relative velocities, is extracted from the state of the rigid bodies. Section 2.5.5 discusses how the normal contact forces are determined using interference volume information. In Section 2.5.6, the tangential friction model is discussed, followed by section 2.5.7, which discusses how the volume of interference is determined using the Muller-Preperata and QuickHull algorithms. Finally, Section 2.5.8 discusses how collisions are detected using the GJK algorithm, including the theory behind MSD and MTD.

2.5.4 Point Contact Model

Given the two rigid bodies shown in Figure 16, evaluating the dynamics of each of these bodies requires knowledge of the direction and magnitude of the contact forces acting on each of these objects. Within the SMS API, the contact force is modeled using a point contact model, which applies a single contact force at a single location on the rigid body. To determine the magnitude of the contact force, point contact models lump the stress distribution over the contact surface and apply it as an equivalent lumped load at the location of the stress distribution centroid as seen in Figure 17. The contact force is applied to the rigid body object at a point in or on the object at some location $\mathbf{C}_\sigma$ away from the body-fixed frame origin. A rigid body, such as in Figure 16, will see the contact forces acting about its body-fixed frame B as:

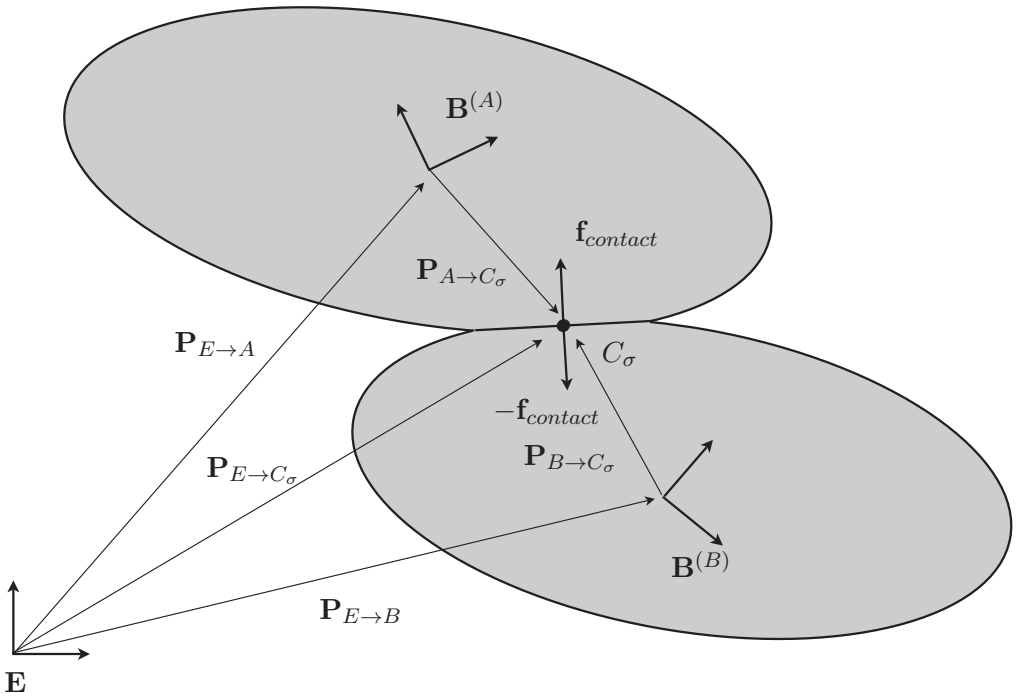


Figure 16: *The interference of two bodies showing their body-fixed frames A and B relative to the Earth-fixed frame E as well as the centroid of contact pressure, $\mathbf{C}_\sigma$, and the contact force, $\mathbf{f}_{contact}$, which is normal to the contact surface.*

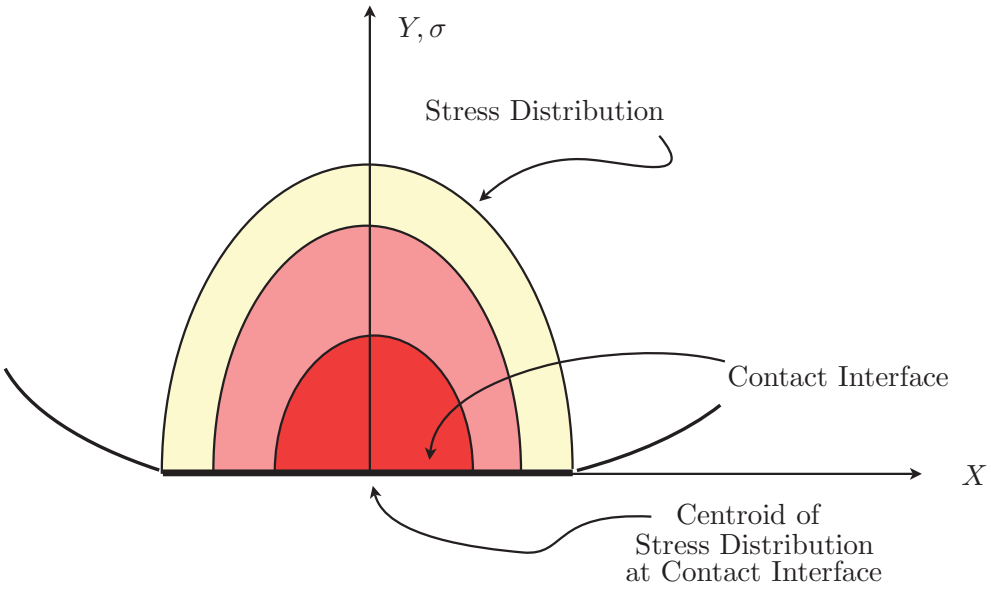


Figure 17: *The stress distribution of a sphere with elastic material properties contacting a non deformable flat plane.*

$$ {}^{B}\mathbf{F}_{contact} = {}^{B}\,\mathbf{T}_{E}\ {}^{E}\mathbf{F}_{contact} \tag{73} $$

where ${}^{B}\mathbf{F}_{contact}$ is the 6×1 spatial contact force vector expressed about the rigid body's body-fixed frame, ${}^{B}\mathbf{T}_{E}$ is a 6×6 transformation matrix that expresses Earth-fixed frame vectors in terms of the body-fixed frame B, and ${}^{E}\mathbf{F}_{contact}$ is the contact force expressed with respect to the Earth-fixed frame:

$$ {}^{E}\mathbf{F}_{contact} = \begin{bmatrix} {}^{E}\mathbf{f}_{contact} \\ {}^{E}\mathbf{P}_{B \to C_\sigma} \times^{E} \mathbf{f}_{contact} \end{bmatrix} \tag{74} $$

where ${}^{E}\mathbf{P}_{B \to C_\sigma}$ is a vector that starts at the rigid body's body-fixed frame and ends at the stress distribution centroid C_σ and is expressed with respect to the Earth-fixed frame, and ${}^{E}\mathbf{f}_{contact}$ is a 3×1 lumped contact force vector applied at $\mathbf{C}_\sigma$ described about the Earth-fixed frame.

The velocity of rigid body A relative to rigid body B at the centroid of the volume of interference is:

$$ {}^{E}\mathbf{v}_{C_\sigma | A-B} = {}^{E}\,\mathbf{T}_{B^{(A)}}\ {}^{B^{(A)}}\mathbf{T}^{(v)}_{B^{(A)} \to C_\sigma}\ {}^{B^{(A)}}\mathbf{v}_{B^{(A)}} - {}^{E}\,\mathbf{T}_{B^{(B)}}\ {}^{B^{(B)}}\mathbf{T}^{(v)}_{B^{(B)} \to C_\sigma}\ {}^{B^{(B)}}\mathbf{v}_{B^{(B)}} \tag{75} $$

where ${}^{E}\mathbf{v}_{C_\sigma|A-B}$ is the velocity of object A relative to object B at the interference volume centroid C_σ, ${}^{B^{(A)}}\mathbf{v}_{B^{(A)}}$ is the velocity of object A expressed in terms of object A's body fixed-frame, ${}^{B^{(B)}}\mathbf{v}_{B^{(B)}}$ is the velocity of object B in terms of object B's body-fixed frame, ${}^{E}\mathbf{T}_{B^{(A)}}$ and ${}^{E}\mathbf{T}_{B^{(B)}}$ are 6×6 spatial transformation matrices that transform a spatial vector's frame of reference from frame $B^{(A)}$ or $B^{(B)}$ to the Earth-fixed frame E, and ${}^{B^{(A)}}\mathbf{T}^{(v)}_{B^{(A)}\to C_\sigma}$ and ${}^{B^{(B)}}\mathbf{T}^{(v)}_{B^{(B)}\to C_\sigma}$ are 6×6 velocity transformation matrices that are used to determine the velocities at C_σ given the velocities at frames $B^{(A)}$ or $B^{(B)}$ (see Equation 37 for a description of a velocity transformation matrix construction).

The tangential relative velocity required to determine the direction of the tangential contact force component is:

$$ {}^{E}\mathbf{v}^{(\hat{t})}_{C_\sigma|A-B} = {}^{E}\mathbf{v}_{C_\sigma|A-B} - {}^{E}\mathbf{v}^{(\hat{n})}_{C_\sigma|A-B} \tag{76} $$

where

$$ {}^{E}\mathbf{v}^{(\hat{n})}_{C_\sigma|A-B} = \frac{\left({}^{E}\mathbf{P}^{(A)}_{(min)} - {}^{E}\mathbf{P}^{(B)}_{(min)}\right)}{\left|{}^{E}\mathbf{P}^{(A)}_{(min)} - {}^{E}\mathbf{P}^{(B)}_{(min)}\right|} \cdot {}^{E}\mathbf{v}_{C_\sigma|A-B} \tag{77} $$

where ${}^{E}\mathbf{v}^{(\hat{t})}_{C_\sigma|A-B}$ is the tangential relative velocity component, ${}^{E}\mathbf{v}^{(\hat{n})}_{C_\sigma|A-B}$ is the normal relative velocity component, and ${}^{E}\mathbf{P}^{(A)}_{(min)}$ and ${}^{E}\mathbf{P}^{(B)}_{(min)}$ are the points on each object associated with the MTD, and are shown in Figure 18.

If a contact event exists between two rigid bodies, the magnitude and direction of the contact force as well as the location on the objects where the force is applied must be determined. To accomplish this, Section 2.5.5 describes the normal contact force model based on volumetric information exhibiting asymmetric damping that is employed. Then, the simple dead-zone Coulomb friction model that is superimposed on the normal contact model to define the tangential forces is described in Section 2.5.6. The location of center of pressure, or the location of the force application point, analogous to the centroid of the volume of interference between the two objects, is described in Section 2.5.7.3. The normal contact force direction is approximated using the MTD, which is determined as described in Section 2.5.8.4.

2.5.5 Normal Contact Force

When two objects come into contact, a normal pressure field is formed between them. This pressure field, seen in Figure 17, is caused by the deformation of the material of

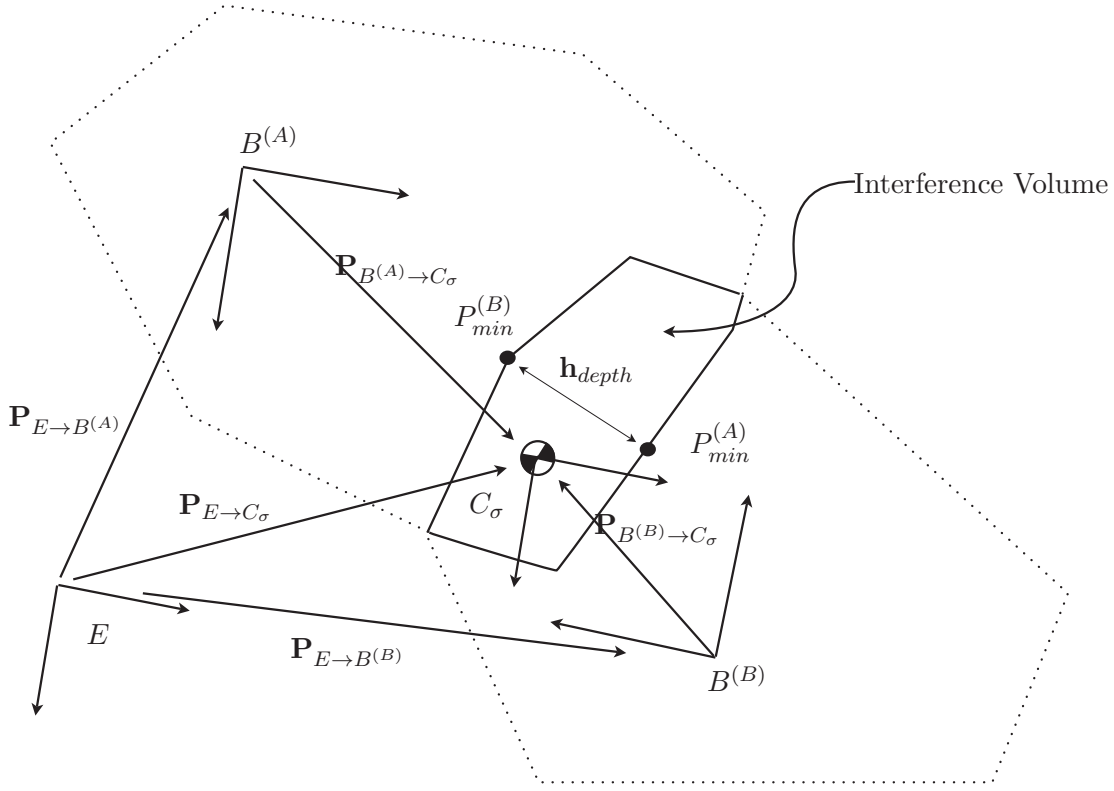


Figure 18: *The volume of interference and its centroid.*

the two contacting objects. Integrating the normal pressure over the contact surface will yield the total normal restitutional force:

$$f_{\hat{n}}^{(s)} = \int \sigma_{\hat{n}}(s) ds \tag{78}$$

where $f_{\hat{n}}^{(s)}$ is the stiffness component of the normal restitutional contact force, $\sigma_{\hat{n}}(s)$ is the local surface pressure, ds is the differential area the pressure is applied to, and s is a point on the contact surface.

The contact pressure is governed by Hooke's law, which maps the relationship between stress and strain via Young's modulus. Here, the total deformations of the object materials are approximated by the interference geometry volume. Consider the Winkler elastic foundation model, shown in Figure 19, where a rigid object contacts an elastic spring bed. Each spring of the bed deflects under the contacting object and imparts a repelling force according to the spring's stiffness. Integrating the force applied on the object by each spring across the contact surface produces the total contact force, in a similar manner as expressed in equation 78. With a differential spring width,

The Displaced Volume of the Springs

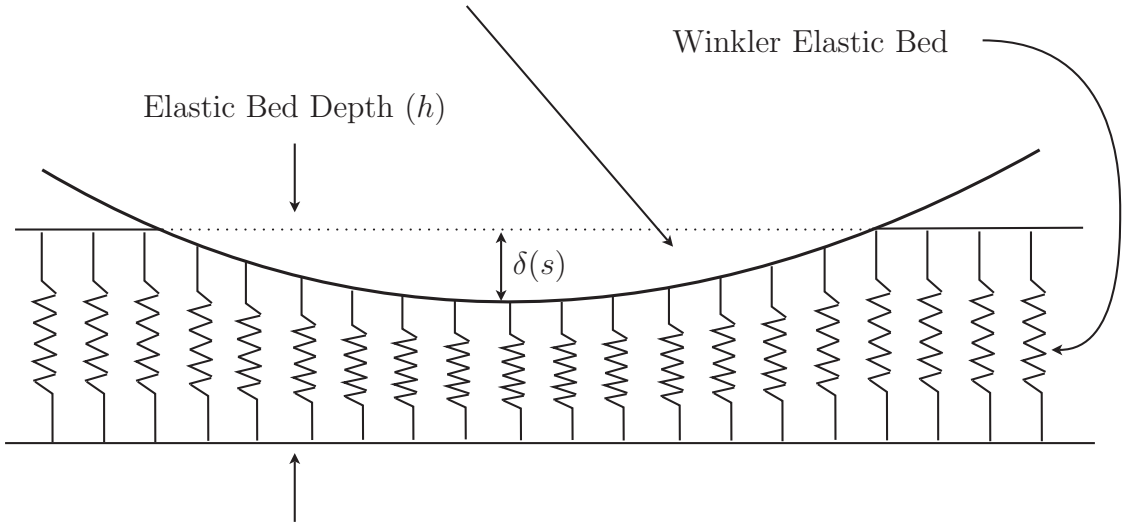


Figure 19: *The Winkler Elastic Foundation model bed showing the local deflections of each spring $\delta(s)$.*

integrating the deflection of each spring across the contact surface produces a displaced volume of the springs.

This is the nature of the link between the volume of interference, analogous to the displaced volume of springs, to strain, stress, and the total contact force. The local stress at a point on the contact surface is defined as:

$$\sigma_{\hat{n}}(s) = E\epsilon_{\hat{n}}(s) \tag{79}$$

where E is the Young's modulus of the object material, and $\epsilon_{\hat{n}}$ is the true strain at a point s on the contact surface. The engineering strain is used as an approximation, and is only valid for small strains:

$$\epsilon_{\hat{n}}(s) \approx e_{\hat{n}}(s) = \frac{\delta(s)}{h} \tag{80}$$

where $e_{\hat{n}}(s)$ is the engineering strain at a point s on the contact surface, $\delta(s)$ is the local deflection of the contact surface and h is the undeflected bed depth.

The volume of interference is analogous to the integral of the local deformations across the contact surface:

$$V = \int \delta(s)ds = \int dV(s) \tag{81}$$

where V is the total displaced volume and $dV(s)$ is the differential volume created by the local deformation multiplied by the differential area. Combining equations 78, 79,

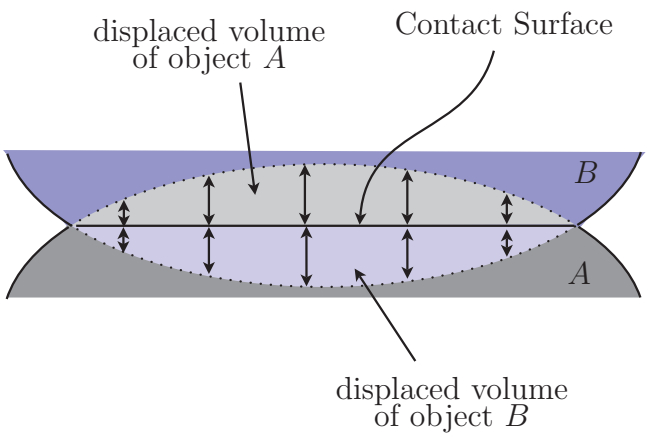


Figure 20: *The displaced volumes of object A and B which form the two parts of the volume of interference.*

80, and 81, a formulation of the contact force based on the displaced volume of an object is:

$$f_{\hat{n}}^{(s)} = \frac{EV}{h} \tag{82}$$

Thus far, only the displaced volume of one object was considered. For real collisions, both colliding objects would deform under the contact pressure. The volume of interference between two objects is equal to the sum of both object's displaced volume as seen in Figure 20. The spring beds of each object are modeled analogously to pairs of springs connected in series, where each spring of object A in contact is modeled in series with its counterpart spring in object B. This provides an effective stiffness coefficient for the system. The contact force applied to both objects becomes:

$$f_{\hat{n}}^{(s)} = \frac{E_A E_B V_{int}}{h_B E_A + h_A E_B} \tag{83}$$

or for the case of objects of identical material and contact geometries (where $E_A = E_B$ and $h_A = h_B$):

$$f_{\hat{n}}^{(s)} = \frac{EV_{int}}{2h} \tag{84}$$

In a similar fashion, an equivalent damping effect can be obtained via the engineering strain rate ($\dot{e} \approx \dot{\varepsilon}$ for small strains):

$$f_{\hat{n}}^{(d)}(s) = B \int \dot{e}_{\hat{n}}^{m}(s) ds \tag{85}$$

where B is the damping factor (strain hardening coefficient) and m is the strain rate sensitivity exponent [38]. The local engineering strain rate, $\dot{e}_{\hat{n}}(s)$, is defined as:

$$\dot{e}_{\hat{n}}(s) = \frac{1}{h}\frac{d(\delta(s))}{dt} \tag{86}$$

where $\frac{d(\delta(s))}{dt}$ is the local deformation rate. When dealing with objects of dissimilar materials, more specifically when strain rate sensitivity of each object, m_A and m_B, differ, it is difficult to obtain the damping component of the contact force independent of the velocity of the contact surface. When $m_A = m_B$, such as when the damping component are linear, as assumed here for simplicity, this is no longer an issue. The total contact force, accounting for different material damping properties and geometries, becomes:

$$f_{\hat{n}} = \frac{(V_{int})E_A E_B}{h_B E_A + h_A E_B} + \frac{B_A B_B}{h_B B_A + h_A B_B}\frac{dV_{int}}{dt} \tag{87}$$

2.5.5.1 Determining the Spring Bed Depth

Engineering strain is a function of both change in length and undeformed length or spring bed depth h from equation 80. The radial strain for a solid sphere is defined as

$$e_r(s) = \frac{\delta R(s)}{R_i} \tag{88}$$

where $\varepsilon_r(s)$ is the radial strain at a location s on its surface, $\delta R(s)$ is the change in radius at a location s on its surface, and R_i is the undeformed radius of the sphere. However, the strain for general objects is more difficult to formulate because the original undeformed length is not necessarily easily defined.

To simplify the problem for general body geometries, the volume of an object is equated to the volume of a sphere through which an equivalent radius can be quantified and used as a reference bed depth. This assumption provides an average reference undeformed length, or spring bed depth, which is required to calculate engineering strain as an approximation of the strain due to contact for that object:

$$R_{eq}^{(A)} = \sqrt[3]{\frac{3V_A}{4\pi}} \tag{89}$$

where $R_{eq}^{(A)}$ is the equivalent radius based on the volume of a sphere for object A, and V_A is the volume of object A.

2.5.6 Tangential Contact Force

The SMS API makes use of the Karnopp friction model, which is a simple dead-zone Coulomb friction model that handles stick/slip transitions. The Coulomb friction force for objects with relative motion of their contact surfaces is:

$$ {}^{B(A)}\mathbf{f}^{(A)(\hat{t})}_{Coulomb} = -\mu_c f_{\hat{n}} \frac{{}^{B(A)}\mathbf{v}^{(\hat{t})}_{C_\sigma|A-B}}{\left|{}^{B(A)}\mathbf{v}^{(\hat{t})}_{C_\sigma|A-B}\right|} \tag{90} $$

where μ_c is the Coulomb fiction coefficient (for sliding friction) and ${}^{B(A)}\mathbf{f}^{(A)(\hat{t})}_{Coulomb}$ is the tangential Coulomb friction force for object A.

When two objects are touching without relative motion and a differential tangential force is applied, a static sticking friction force is present preventing tangential acceleration. This force is greater than the dynamic sliding Coulomb force. The differential tangential force required to overcome sticking friction is:

$$ f^{(\hat{t})}_{stick} = \mu_s f_{\hat{n}} \tag{91} $$

where μ_s is the sticking friction coefficient and $f^{(\hat{t})}_{stick}$ is the magnitude of the force that the sticking friction force will oppose up to the point of break-away, which is the initiation of sliding motion.

Using the Karnopp model to account for the sticking and sliding transitions, the friction force is modeled as:

$$ {}^{B(A)}\mathbf{f}^{(A)(\hat{t})}_{fric} = \begin{cases} {}^{B(A)}\mathbf{f}^{(A)(\hat{t})}_{Coulomb} & if \left|{}^{B(A)}\mathbf{v}^{\hat{t}}_{C_\sigma|A\to B}\right| > v_{tol} \\ -{}^{B(A)}\mathbf{f}^{(A)(\hat{t})} & if\ \left|{}^{B(A)}\mathbf{v}^{\hat{t}}_{C_\sigma|A\to B}\right| < v_{tol}\ \&\ \left|{}^{B(A)}\mathbf{f}^{(A)(\hat{t})}\right| < f^{(\hat{t})}_{stick} \\ -\frac{{}^{B(A)}\mathbf{f}^{(A)(\hat{t})}}{\left|{}^{B(A)}\mathbf{f}^{(A)(\hat{t})}\right|} f^{(\hat{t})}_{stick} & if\ \left|{}^{B(A)}\mathbf{v}^{\hat{t}}_{C_\sigma|A\to B}\right| < v_{tol}\ \&\ \left|{}^{B(A)}\mathbf{f}^{(A)(\hat{t})}\right| > f^{(\hat{t})}_{stick} \end{cases} \tag{92} $$

where ${}^{B(A)}\mathbf{f}^{(A)(\hat{t})}_{fric}$ is the friction force felt by object A, v_{tol} is the velocity magnitude tolerance limit defining the dead-zone, and ${}^{B(A)}\mathbf{f}^{(A)(\hat{t})}$ is the tangential component of the forces acting on object A, which will be countered by the sticking friction force that results in no tangential acceleration. When the relative velocity between the objects is less than v_{tol}, the sticking friction mode will match any forces tangential to the contact surface, applied to objects A and B up to the break-away force $f^{(\hat{t})}_{stick}$.

2.5.7 Volume of Interference and Other Interference Properties

The most computationally intensive portion of volumetric contact models is typically computing the volume of interference. For simple geometries, such as spheres and

cylinders, analytical solutions to the volume of interference exist that make computing volumes efficient [35]. For convex polyhedra, a $O((N_A + N_B)log(N_A + N_B))$ algorithm was suggested for use where N_A and N_B are the number of facets each convex polyhedral object has [28]. The Muller-Preparata algorithm (MPA) is especially well suited for use with GJK because it requires knowledge of a point known to be contained within both interfering objects [39]. GJK can quickly provide the point on each object that together represents the penetration depth, $\mathbf{P}_A^{(min)}$ and $\mathbf{P}_B^{(min)}$, as described in Section 2.5.8.4. A point taken halfway between $\mathbf{P}_A^{(min)}$ and $\mathbf{P}_B^{(min)}$ is guaranteed to be located within both interfering geometries.

One of the steps of the MPA is to calculate the convex hull of a set of vertices. To accomplish this, the Quickhull algorithm, so-called for its similarity to the QuickSort algorithm, is used because it is among the most efficient convex hull algorithms with a complexity of $O(NlogN)$, where N is the number of vertices [1, 28, 40]. The QuickHull algorithm is discussed in section 2.5.7.1, which is followed by a description of the MPA algorithm in Section 2.5.7.2.

2.5.7.1 Quickhull: An Efficient Algorithm to find the Convex Hull of a set of Points

A convex hull of a set of points in space is the smallest convex volume that can be created to encapsulate the set of points. A popular analogy of the convex hull for planar objects is to visualize an elastic band stretched around pegs on a board (the vertices) and release it, thus allowing it to assume the minimum convex shape that encapsulates all vertices. This is illustrated in Figure 21.

The Quickhull algorithm takes a set of vertices and finds its convex hull polyhedral representation. A detailed description of the algorithm is provided in the literature [1]. In 3D, Quickhull requires at least a non-degenerate tetrahedron to begin. The tetrahedron, a polyhedron, consists of 4 vertices and 4 facets. The Quickhull algorithm now consists of a tetrahedron and the remaining set of vertices. The algorithm is prepared by placing every remaining vertex from the original set of vertices in a set of unassigned vertices. Every facet of the tetrahedron is then checked against the current set of unassigned vertices, picking out any vertex that lies in the positive direction of the facet normal relative to a point on that face. These points are assigned to the facet's set of outside vertices, and the facet is placed in a list of facets with an unempty outside vertex set. The hull is formed when there are no facets left with points outside.

The algorithm begins by taking the first face with an unempty outside set and identifying the furthest vertex in the set. If a vertex lies outside a face, then that face is not part of the convex hull of the original set of vertices. Therefore, that facet is added to a list of facets visible to the furthest vertex, which will later be removed from

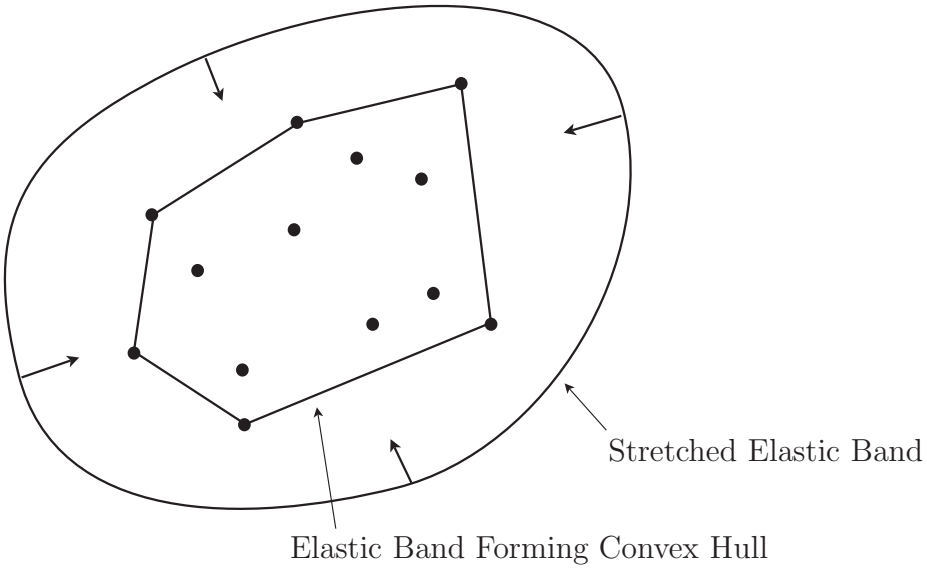


Figure 21: *The elastic band analogy of the convex hull.*

the polyhedron. The neighbouring faces are then checked to see if the furthest vertex lies outside those faces as well, and thus adding them to the list of visible facets. This process of checking if the furthest vertex lies outside every neighbouring face is repeated for all the faces in the expanding list of facets visible to the furthest vertex until no more visible neighbouring facets can be found.

All edges for a closed polyhedron should be shared by two faces. The edges with a facet in the visible set of facets and the other facet not in the visible set are added to a list of edges that will form a ridge when the facets in the list of visible facets are removed from the polyhedron. First a new facet is created with each of the ridge edges using the furthest vertex as the 3rd vertex and adding the new facets to the polyhedron, updating the adjacenct matrices. Each newly created facet then checks the set of outside vertices of every facet in the list of visible facets and assigns vertices outside the new facet to its set of outside vertices. Finally the facets in the list of visible facets are deleted.

This process continues with the next facet in the expanding list of facets with an unempty outside set until there are no longer any facets with an unempty outside set, guaranteeing the final polyhedron is the convex hull of the original set of vertices. Pseudocode for the algorithm is listed in Figure 22.

```
create a non-degenerate tetrahedron (simplex) of 4 vertices and 4 facets

for each facet F
    for each unassigned point p
        if p is in the positive normal direction of F
            assign p to F's outside set

for each facet F with a non-empty outside set
    select the furthest point p of F's outside set
    initialize the visible set V to F
    for all unvisited neighbors N of facets in V
        if p is above N
            add N to V
    the boundary of V is the set of horizon ridges H
    for each edge E in H
        create a new facet from E and p
        link the new facet to its neighbours
    for each new facet F'
        for each unassigned point q in an outside set of a facet in V
            if q is above F'
                assign q to F''s outside set
    delete the facets in V
```

Figure 22: *The pseudocode for Quickhull [1]*

2.5.7.2 Muller-Preparata Interference Polyhedron Algorithm

The Muller-Preparata algorithm (MPA) is a simple algorithm that finds the geometry of the common space shared by two convex polyhedra, their interference geometry [28, 39]. This algorithm runs in $O((N_A + N_B)log(N_A + N_B))$ time where N_A and N_B are the number of facets belonging to the convex polyhedral objects A and B respectively.

The algorithm makes use of geometric dual transformations [28]. The dual transformation converts a plane whose equation is described by $ax+by+cz = 1$ into a point (a, b, c) or alternatively it converts a point (a, b, c) into a plane of $ax + by + cz = 1$ assuming a point common to both objects, located in the interference geometry, is taken as the origin. The MPA works as follows:

1. Consider a point in common to both polyhedra as the origin.
2. Create a new set of vertices by converting the planes of each polygon of both objects using dual transformation.
3. Find the convex hull of the new set of vertices from step 2.
4. Create a new set of vertices by converting the planes of the convex hull from step 3 using dual transformation.
5. Find the convex hull of the new set of vertices from step 4.

The end results of the algorithm, after step 5, is a polyhedron that encapsulates the common space between both interfering polyhedra, which is the interference geometry.

The convex polyhedron returned by this algorithm is the exact interference geometry between the two objects, within what machine tolerance allows. Because of machine tolerance issues, the SMS implementation also uses a zero distance tolerance of 1 micron, that is a vertex is considered the same vertex or considered to lie on a plane if it lies within 1 micron of the other. This adds the potential for some minor error.

2.5.7.3 Volume and Centroid of a Polyhedron

The volume of a polyhedron, convex or concave, can be obtained by summing up the volumes of the tetrahedron created by each triangular polygon facet of the polyhedron and the origin [41]. The volume of a tetrahedron is defined using the scalar triple product:

$$V_{tetra} = \frac{|(\mathbf{v}_1 - \mathbf{v}_4) \cdot (\mathbf{v}_2 - \mathbf{v}_4) \times (\mathbf{v}_3 - \mathbf{v}_4)|}{6} \tag{93}$$

where $\mathbf{v}_1$, $\mathbf{v}_2$, $\mathbf{v}_3$ and $\mathbf{v}_4$ are the vertices of the tetrahedron. Alternatively, by setting $\mathbf{v}_4$ to the origin, the volume of a tetrahedron formed by a polygon facet of a polyhedron and the origin can be obtained as:

$$V_{tetra} = \frac{\mathbf{v}_1 \cdot \mathbf{v}_2 \times \mathbf{v}_3}{6} \tag{94}$$

This allows the summing all of the volumes, V_{tetra}, of the tetrahedrons created by the vertices of the polygons of a polyhedron and the origin and will yield the total volume of the polyhedron. The volume V_{tetra} is signed and depending on which side of the polygon the origin lies on, can produce positive or negative volumes. This yields a net volume equal to the volume of the polyhedron. The volume cancelling nature of this algorithm also works well for finding the volumes of concave objects, though these are not considered here.

The location of the centroid of these tetrahedra are found as [41]:

$$\mathbf{P}_{centroid} = \frac{\mathbf{v}_1 + \mathbf{v}_2 + \mathbf{v}_3}{4} \tag{95}$$

The center of gravity of a polyhedron can be found by summing the tetrahedral centroids weighted by their signed volumes, and dividing the sum with the total polyhedron volume.

2.5.8 Detecting Collisions of Convex Polyhedra with GJK

To determine the contact forces discussed above, the collision must first be detected. The GJK algorithm and Minkowski difference properties are exploited to this end. Section 2.5.8.1 describes the Minkowski difference of two convex objects, which has useful properties including the fact that if the origin is located within the Minkowski difference space, the two objects are guaranteed to be colliding. Then, Section 2.5.8.2 describes how the properties of the Minkowski difference are used to determine if a collision is occuring. Section 2.5.8.4 goes on to describe how GJK is used to obtain the MSD or MTD using barycentric coordinates.

2.5.8.1 Minkowski Difference

The Minkowski sum is defined as the sum of every point in a set of points A added with every point in a different set of points B:

$$A \oplus B = \{\mathbf{P}_A + \mathbf{P}_B : \mathbf{P}_A \in A, \mathbf{P}_B \in B\} \tag{96}$$

Geometrically, the Minkowski sum between two objects can be visualized as the area covered by picking up object B by an imaginary handle at the origin, placing the handle

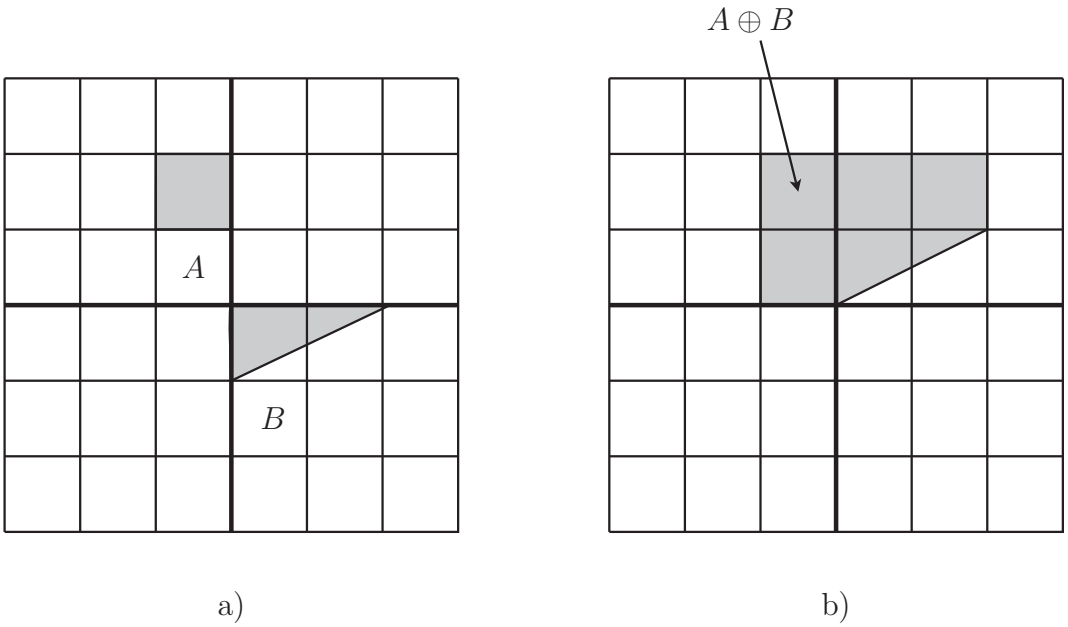


Figure 23: *An example of the Minkowski sum between two objects showing a) objects A and B and b) their Minkowski sum.*

in the space occupied by A, and sweeping the handle such that all the space occupied by A is covered by the handle. The swept space covered by B during the translation process represents the Minkowski sum as indicated in Figure 23.

The Minkowski difference, $A \ominus B$, is simply the Minkowski sum of object A with the mirror of object B about the X, Y, and Z axes as indicated in Figure 24. In other words:

$$A \ominus B = A \oplus (-B) \tag{97}$$

The Minkowski difference of convex objects has a few interesting properties not necessarily present for concave objects, which include:

- The Minkowski difference encapsulates the origin only when the objects of the Minkowski difference are interfering.
- The minimum separation distance in real space is represented in the Minkowski difference space as the minimum separation distance between the Minkowski difference and the origin as illustrated in Figure 25.

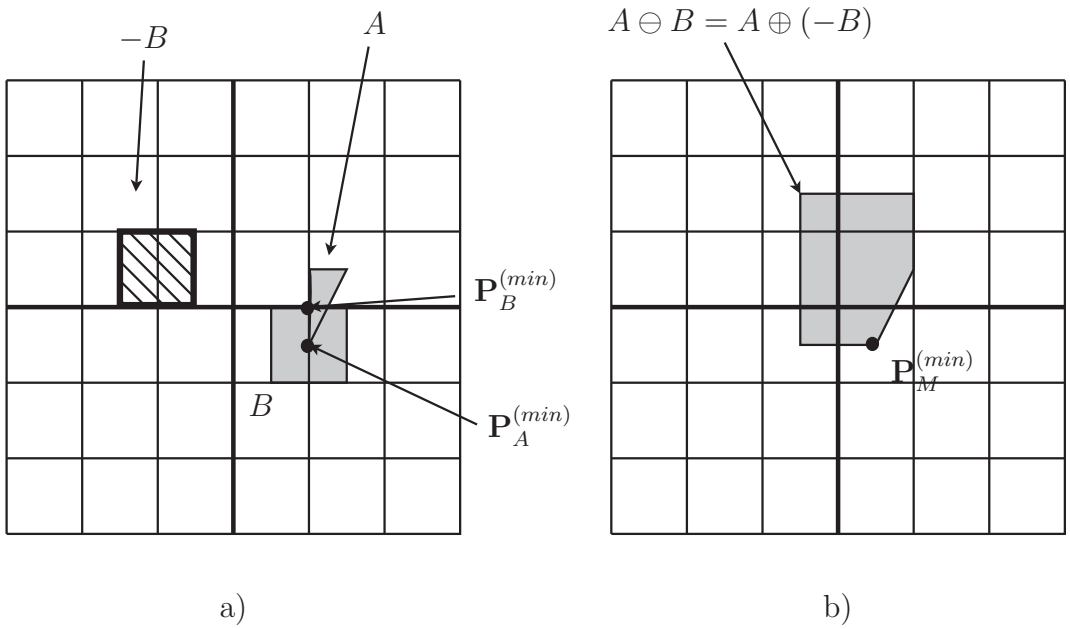


Figure 24: *An example of the Minkowski difference between two objects showing a) objects A and B and -B b) the Minkowski difference between A and B.*

- The MTD is defined as the distance between the origin and the closest point on the Minkowski difference boundaries.

If the point of minimum separation between the boundaries of the Minkowski difference and the origin is known $\mathbf{P}_M^{(min)}$, it is possible to determine which points on the surfaces of objects A and B that minimum separation corresponds to, $\mathbf{P}_A^{(min)}$ and $\mathbf{P}_B^{(min)}$, using barycentric coordinates [2]. This will be discussed in Section 2.5.8.4. The Minkowski difference for convex polyhedra can be obtained by brute force by taking the Minkowski difference between the set of vertices of both objects. The convex hull that encapsulates all points of the Minkowski difference can easily be found as described in Section 2.5.7.1.

2.5.8.2 GJK

The Gilbert-Johnson-Keerthi (GJK) algorithm is one of the most effective methods of detecting collisions between convex polyhedral objects [2]. It is based on the properties of the Minkowski difference though the explicit calculation of the Minkowski difference is not required. Instead the GJK is a descent algorithm that samples the Minkowski difference using a support mapping function, discussed in Section 2.5.8.3. GJK is independent of geometrical representation of the objects so long as a support function can be provided. The GJK algorithm is especially effective with polyhedral geometry

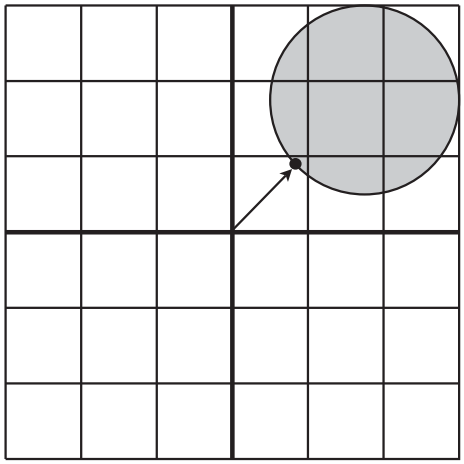


Figure 25: *The minimum separation distance between the Minkowski difference space and the origin.*

representation because the piece-wise linearised nature of the boundary ensures exact solutions are converged on quickly compared to objects with continuous surfaces that require a tolerance factor to check for convergence. A 3D implementation of the algorithm has been shown to have a computational complexity of $O(N_A + N_B)$ where N_A and N_B are the total number of vertices of objects, A and B, respectively [23].

In 3D, GJK attempts to find a 3-simplex from the Minkowski difference that encapsulates the origin of the volume of interference, where an n-simplex is the n-dimensional convex hull of a set of $n + 1$ points. This would establish that a collision is occurring. The algorithm starts with a 0-simplex, and using the support function, finds the furthest Minkowski difference vertex pair in the direction $\mathbf{d}$. The direction $\mathbf{d}$ is chosen as $-\mathbf{P}_M^{(min)}$, which tells the support mapping function to find the furthest vertex in the direction of the origin where $-\mathbf{P}_M^{(min)}$ is a point on the current n-simplex that is closest to the origin, as indicated in Figure 26. This support function has the convenient property that:

$$S(A \oplus B, \mathbf{d}) = S(A, \mathbf{d}) - S(B, -\mathbf{d}) \tag{98}$$

that makes finding Minkowski difference vertices simple, where $S(A, \mathbf{d})$ and $S(B, \mathbf{d})$ are support functions that return the furthest vertex in the direction of vector $\mathbf{d}$ (or $-\mathbf{d}$) from objects A and B, respectively. Obtaining the furthest vertex of a polyhedron in a particular direction requires at most $O(N)$ distance checks, where N is the number of vertices in the polyhedron; however, with some pre-processing, such as building a

vertex adjacency matrix, this can be reduced to $O(logN)$ checks. Thus finding the furthest vertex of the Minkowski difference in any direction requires on the order of $O(logN_A + logN_B)$ distance computations.

In 3D, GJK works by maintaining a set Q of at least 1 and up to 4 vertices whose convex hull forms a 0-simplex, a 1-simplex, a 2-simplex, or a 3-simplex depending on the number of points held in Q. If the origin is contained within the current simplex, *i.e.* the origin is at a vertex, on an edge, on a facet or inside the tetrahedron, then objects A and B are intersecting. If this is determined to be true, the algorithm stops and returns that simplex. On the other hand, if the simplex does not contain the origin, the simplex is updated to the smallest subset of Q that forms a feature (*i.e.,* a vertex, an edge or a polygon) that contains the point on that feature that is closest to the origin, $\mathbf{P}_M^{(min)}$. A new vertex is found from $A \ominus B$, in the direction of $\mathbf{d}$. Repeating the cycle of reducing the simplex and finding furthest vertices along $\mathbf{d}$ guarantees that eventually the simplex will contain either the origin or no further vertex in $\mathbf{d}$ will be found, ending the search.

The GJK loop begins with a Q forming a 0-simplex consisting of a single point, Q_1. A new test point, $Q_{furthest}$, is found in the direction of $-\mathbf{P}_M^{(min)}$, which is the same as $-Q_1$ for the case of a 0-simplex. If $Q_{furthest}$ is further in $\mathbf{d}$ than $\mathbf{P}_M^{(min)}$ then $Q_{furthest}$ is added to Q as the new Q_1, pushing the old Q_1 to Q_2 and so on and forming a new higher dimensional simplex. The feature of the new simplex whose Voronoi region holds the origin is identified, where a feature's Voronoi region is a region in space where the feature is guaranteed to be closest to any points in said region. The Voronoi region is illustrated in 2D in Figure 27. Any vertex in Q not belonging to the identified closest feature is removed from the set Q. A new direction is set as $\mathbf{d} = -\mathbf{P}_M^{(min)}$, and the algorithm loops back to finding $Q_{furthest}$. The loop is broken if no vertex further in $\mathbf{d}$ can be found, or if the origin is encapsulated by the simplex. An example in 2D of how GJK operates is provided in Figure 26.

2.5.8.3 Efficient GJK Support Mapping Function for Convex Polyhedra

Finding the furthest vertex in any particular direction is as simple as checking the distance of every vertex and keeping track of the furthest found. However, this brute force method has a complexity of $O(n)$ where n is the number of vertices. For convex polyhedra, a vertex adjacency matrix can be created that provides each vertex with knowledge of the vertices it shares one or more features with. With a vertex adjacency matrix, the furthest vertex can be found by comparing its distance with those of its neighbors, repeating the process with the furthest vertex found until no adjacent vertex is found to be any further. This leads to a more efficient search complexity of $O(logn)$.

Vertex $\mathbf{v}_a$ is said to be further in a particular direction, d, than vertex $\mathbf{v}_b$, if the dot

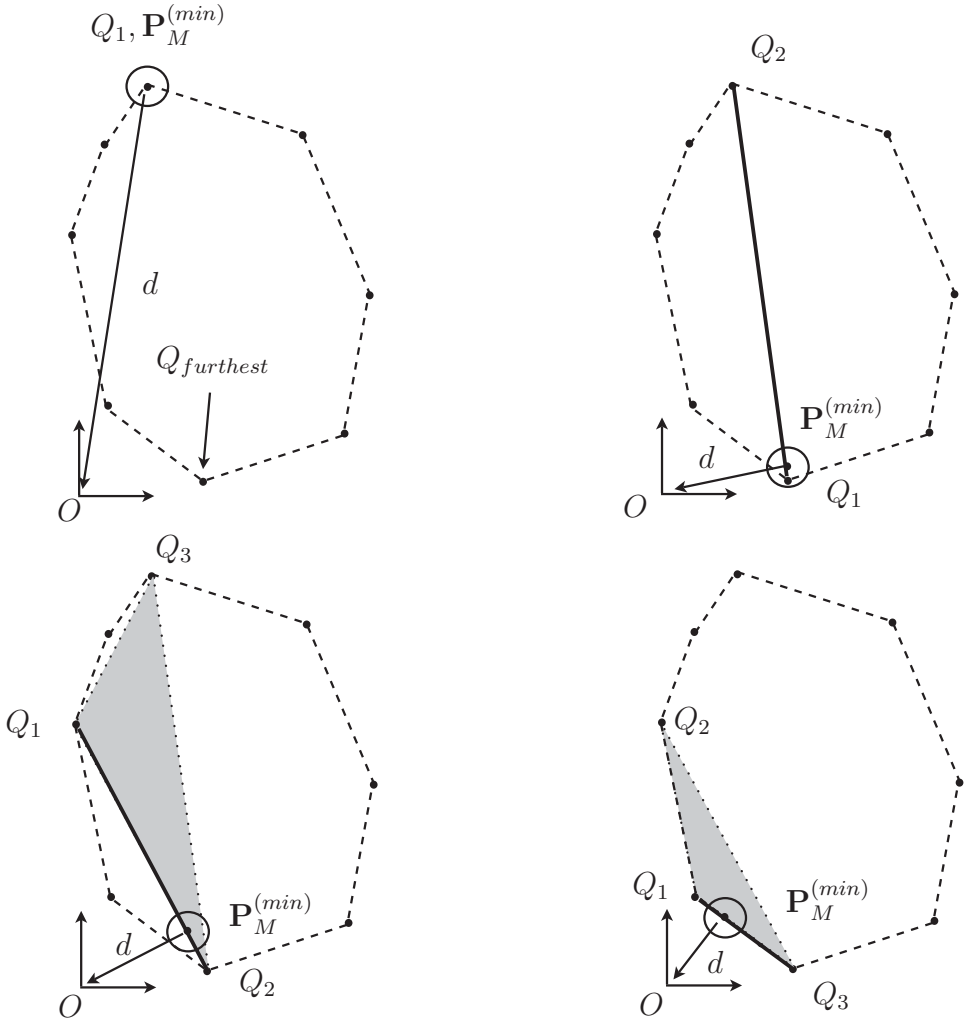


Figure 26: *The GJK algorithm finding the closest point to the origin on the Minkowski difference, in 2D for simplicity. This example was based on an example from [2].*

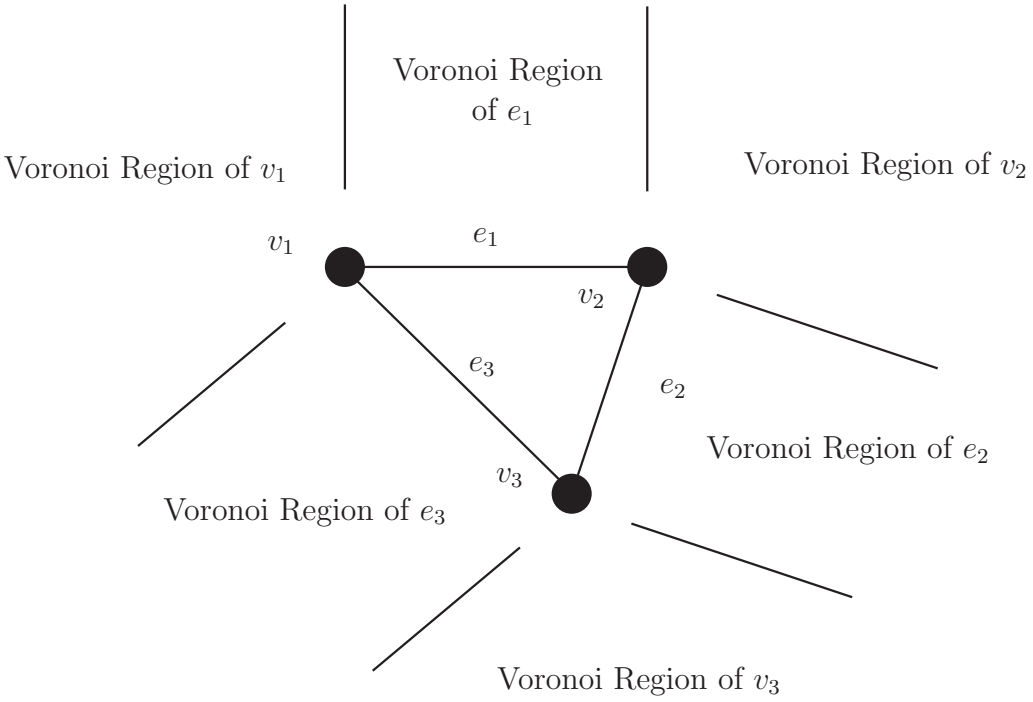


Figure 27: *A 2D example of the Voronoi region for a polygonal object, identifying the Voronoi regions for each edge, e_i, and each vertex, v_i.*

product $\mathbf{v}_a \cdot \mathbf{d}$ is greater than $\mathbf{v}_b \cdot \mathbf{d}$.

2.5.8.4 The Minimum Separation Distance/Minimum Translation Distance using Barycentric Coordinates

A point on a polygon of the Minkowski difference $A \ominus B$, $\mathbf{P}_M^{(min)}$ for example, can be mapped back to its corresponding pair of points, one on object A and one on object B. That is, for the 3D case, the barycentric coordinates, $[c_1, c_2, c_3]$, of $\mathbf{P}_M^{(min)}$ on a 2-simplex, can be defined as:

$$\mathbf{P}_M^{(min)} = \sum_{i=1,2,3} c_i \mathbf{Q}_i \tag{99}$$

where

$$\sum_{1 \leq i \leq 3} c_i = 1 \tag{100}$$

and

$$c_i \geq 0 \tag{101}$$

Because the points on the Minkowski difference consist of a point on object A subtracted from a point object B, we can split $\mathbf{P}_M^{(min)}$ into its components:

$$\mathbf{P}_M^{(min)} = \mathbf{P}_A^{(min)} - \mathbf{P}_B^{(min)} \tag{102}$$

If knowledge of the points on objects A_i and B_i that created the point Q_i of equation 99 is retained, $\mathbf{P}_A^{(min)}$ and $\mathbf{P}_B^{(min)}$ can be mapped from $\mathbf{P}_M^{(min)}$ using the barycentric coordinates:

$$\mathbf{P}_A^{(min)} = \sum_{i=1,2,3} c_i \mathbf{v}^{(\mathbf{A})}{}_i \tag{103}$$

and

$$\mathbf{P}_B^{(min)} = \sum_{i=1,2,3} c_i \mathbf{v}^{(\mathbf{B})}{}_i \tag{104}$$

where $\mathbf{v}_i^{(A)}$ and $\mathbf{v}_i^{(B)}$ are the vertices of A and B corresponding to the vertices of Q. The barycentric coordinates are easily solved for by converting equation 99 to matrix form, and solving the equation for the barycentric coordinates.

3 Software Reference Manual

The following section serves as a reference manual for the SMS API software. Instructions on how to build the DLL from source, how to install the required Boost libraries, as well as how to use the DLL in a Visual C++ project are provided in Sections 3.1.1, 3.1.2, and 3.1.3, respectively. Following this, the initialization file specifications for the simulation objects are provided in section 3.2. It should be noted that while many properties are constant, some of the properties set in the initialization files can be modified during simulation execution. Important simulation object class methods are described in section 3.3, followed by the state vector definition for each class in Section 3.4, and the simulation object class output file formats in Section 3.5.

A validation test suite was included as part of the submission of this report and SMS API. The validation test suite consists of a set of sample simulations that make use of the SMS API. The validation test suite offers multiple simple examples of how each of the simulation objects are used together or by themselves in addition to highlighting the accuracy of the system with comparisons to analytical results when applicable.

3.1 SMS API DLL

The SMS API was written in ANSI C++. It has been tested using Microsoft Visual Studio 2008 Express and the bundled Microsoft compiler.

3.1.1 Setting up a project to build the DLL

The following steps are required to set up a VC++ project and build the SMS API DLL:

1. In Visual C++ 2008, click on *File -> New -> New Project* which will open up the *New Project* Dialog.
2. In the *Project Types* Pane, select *Win32* and in the *Templates* pane select *Win32 Console Application*. Name the project "SMS" and select a folder to store its files in and click *OK*.
3. A Win32 Application wizard will appear, Click Next.
4. Select *DLL* for *Application Type* and check off *Empty project* then click *Finish*.
5. Add the SMS API source files to the project that has just been created.

3.1.2 Boost Library Dependency

The SMS API DLL requires the Boost C++ libraries. The Boost C++ Libraries are a collection of peer-reviewed, open-source, and cross-platform libraries. The Boost C++ Libraries are used for the uBLAS library which provides vector and matrix classes and operators (scaling, addition, multiplication, etc.) that allow safe and efficient linear algebra operations in software.

At the time of writing this report, the Boost headers-only libraries are required. These Boost Libraries can be obtained from:

```
http://sourceforge.net/projects/boost/files/boost/1.44.0/
```

Whichever archive (`.7z`, `.zip`, etc.) is downloaded, the contents can be extracted whereever is convenient. A permanent location is recommeneded, such as:

```
C:\Program Files\boost
```

The archive contains a folder called `boost_1_44_0`, which will be the Boost "root" directory.

To use Boost with Microsoft Visual C++ 2008, the IDE must know where to find the Boost header files. These changes are applied at the IDE level. If the name of your project is "SMS_Sample_App", then:

1. Right-click *SMS_Sample_APP* in the Solution Explorer pane of MSVC++ and select *Properties* from the resulting pop-up menu.
2. Click *Configuration Properties* *C/C++* *General* *Additional Include Directories*
3. Enter the path to the Boost root directory, for example:
   `C:\Program Files\boost\boost_1_44_0`

All of the Boost headers are located within the

```
C:\Program Files\boost\boost_1_44_0\boost\
```

directory. The SMS API includes the uBLAS libraries individually using include calls such as:

```
#include "boost/numeric/ublas/vector.hpp"
```

for the vector class.

3.1.3 Using the Compiled SMS API Binary (dll) in Projects

In order to use the SMS API in applications, the SMS API header files, the compiled .dll and .lib files are required. Add all of the `.h` header files to the project folder, with directory structure. Place the `SMS.dll` and the `SMS.lib` files into the project folder, include the SMS.h header file in your code and compile the project.

3.2 Class Initilialization Files

The properties of each simulation object in the SMS API are loaded via an initialization file that is passed to the object constructor. In the following sections, first a description of the `SMS::Environment` and `SMS::Integrator` initialization files, which are required by the SMS API for any simulations, are provided. Next, the initialization file specifications are described for each simulation object. A sample initialization file is also provided for each simulation object.

3.2.1 Environment Initialization Files

SMS API simulations share several global simulation parameters. These are set in the `env.ini` and `RK45.ini` files; these files must be defined for all SMS API simulations. These files define the environmental properties and the settings used in the integrator. The SMS API looks for these in the root simulation directory. The environmental properties set in the env.ini files are air and water density and kinematic viscosity, and gravity.

Sample env.ini File

```
$airDensity 1.20
$airKinematicViscosity 1.5e-5
$waterDensity 1025.0
$waterKinematicViscosity 1.787e-6
$g 9.81
```

3.2.2 Integrator Initialization File Specification

The Runge-Kutta 4(5) integrator settings are the initial time step, the minimum and maximum allowable time step.

Sample RK45.ini File

```
$Timestep_MAX 1.0
$Timestep_MIN 1e-15
$Timestep 0.0001
```

3.2.3 Cable Initialization File Specification

The following table provides a description of each entry of an `SMS::Cable .ini` file:

Variable Name	Variable Description
$EA	Axial Stiffness (N)
$EI	Bending Stiffness (Nm^2)
$GJ	Torsional Stiffness (Nm^2)
$rho_cable	Density $\left(\frac{kg}{m^3}\right)$
$dia_cable	Cable diameter (m)
$CDc	Cable drag coefficient
$CAc	Added mass coefficient
$CID	Axial damping coefficient
$BCID	Bending damping coefficient
$CE	Compressive elasticity (set to 0 for chain behavior)
$ElementLengthMax	The maximum element length before a node is inserted (m)
$ElementLengthMin	The minimum element length before a node is removed (m)
$NODE0_STATIC	A boolean value that forces the cable's node 0 to be statically attached in space. If the node is static, other objects cannot be attached.
$NODEN_STATIC	A boolean value that forces the cable's node N to be statically attached in space. If the node is static, other objects cannot be attached.
$0|IC	$\dot{X}$ (m/s) X (m) $\dot{Y}$ (m/s) Y (m) $\dot{Z}$ (m/s) Z (m) - The first node state is defined by its position and velocity in the global coordinate system. Multiple nodes are defined by creating multiple entries: $1\|IC ,. . . ,$N|IC
$ElementLengths	The unstretched element lengths (m)
$OutputFolder	The name of folder where the cable results will be stored
$0|ExtMasses	Arclength (m) Diameter (m) Density $\left(\frac{kg}{m^3}\right)$ - Spherical external mass attached to the cable at specified arclength from node 0. Multiple masses can be attached.

Table 2: *Cable class init file description*

Sample Cable Init File

Here is the contents of a sample init file:

```
$EA 4.0e6
$EI 0
$GJ 0
$rho_cable 500.0
$dia_cable 0.024
$CDc 0
$CAc 0
$CID 1e3
$BCID 0
$CE 1

$ElementLengthMax 12
$ElementLengthMin 1

$NODE0_STATIC 0
$NODEN_STATIC 0

$0|IC 0 3.645 0 0 0 15.642
$1|IC 0 3.645 0 0 0 8
$2|IC 0 3.645 0 0 0 1

$ElementLengths 7.642 7

$OutputFolder cable_output_folder
```

3.2.4 Payload Initialization File Specification

The following table provides a description of each entry of an `SMS::Payload .ini` file:

Variable Name	Variable Description
$0|IC	Initial X Velocity (m/s)
$1|IC	Initial Y Velocity (m/s)
$2|IC	Initial Z Velocity (m/s)
$3|IC	Initial $\dot{\phi}$ (rad/s)
$4|IC	Initial $\dot{\theta}$ (rad/s)
$5|IC	Initial $\dot{\psi}$ (rad/s)
$6|IC	Initial X Position (m)
$7|IC	Initial Y Position (m)
$8|IC	Initial Z Position (m)
$9|IC	Initial ϕ (Rad)
$10|IC	Initial θ (Rad)
$11|IC	Initial ψ (Rad)
$0|Mass	The first row of the 6×6 mass-inertia matrix for the Payload
$1|Mass	The second row of the 6×6 mass-inertia matrix for the Payload
$2|Mass	The third row of the 6×6 mass-inertia matrix for the Payload
$3|Mass	The fourth row of the 6×6 mass-inertia matrix for the Payload
$4|Mass	The fifth row of the 6×6 mass-inertia matrix for the Payload
$5|Mass	The sixth row of the 6×6 mass-inertia matrix for the Payload
$Young_modulus	Elastic material property for contact dynamics (Pa)
$B	Material damping property for contact dynamics (Pa·s) y
$muc	Coulomb Friction Coefficient for contact dynamics
$mus	Sticking Friction Coefficient for contact dynamics
$OutputFolder	Specifies the folder name that will hold the simulation results output for this payload

Table 3: *Payload Class init file description*

Sample Payload Init File

Here is the contents of a sample init file:

```
$0|IC 0
$1|IC 0
$2|IC 0
$3|IC 0
$4|IC 0
$5|IC 0
$6|IC 3.645
$7|IC 0
$8|IC -0.346
$9|IC 0
$10|IC 0
$11|IC 0

$0|Mass 7800 0 0 0 0 0
$1|Mass 0 7800 0 0 0 0
$2|Mass 0 0 7800 0 0 0
$3|Mass 0 0 0 500 0 0
$4|Mass 0 0 0 0 500 0
$5|Mass 0 0 0 0 0 500

$Young_modulus  95e7 //Pa
$B 5e7   //Pa-s

$muc 0.05 //Coulomb Friction Coefficient
$mus 0.07 //Sticking Friction Coefficient

$OutputFolder payload_output_folder
```

3.2.5 Boomcrane Initialization File Specification

The following table provides a description of each entry of an `SMS::Boomcrane .ini` file:

Variable	Description
$VERBOSE	A flag to tell the boomcrane to output debug information.
$*id*|LinkDHParameters	D-H Parameters for link *id*. Parameter order: α, a, d, θ
$*id*|LinkJointVariable	Which D-H parameter is variable for Link *id*.
$*id*|LinkCGLocation	The location of the center of gravity for Link *id*'s rigidbody frame relative to its proximal joint frame.
$*id*|LinkCGOrientation	The orientation of the Link *id*'s rigidbody frame to its proximal joint frame.
$*id*|LinkMass	Mass of Link *id*
$*id*|LinkMomentOfIntertia	The Moment of Inertia of Link *id* with respect to Link *id* frame.
$*id*|JointControlType	The joint control type, set to "velocity", "position" or "passive"
$*id*|JointProportionalCoefficient	The proportional gain of the PID joint controller (N/m or N·m/rad)).
$*id*|JointIntegralCoefficient	Integral gain of the PID joint controller (N/(m·s) or Nm/(rad·s)).
$*id*|JointDerivativeCoefficient	Derivative gain of the PID joint controller (N·s/m or N·m·s/rad).
$*id*|MaximumActuatablePower	The maximum power the joint controller is limited to.
$*id*|UpstreamJointLimits	Sets the joint limits for Link *id*'s upstream joint (m for prismatic and rad for revolute joints).
$*id*|JointLimitStiffness	The maximum force the joint is capable of applying (N for prismatic joints, N·m for revolute joints).
$*id*|JointLimitDamping	The velocity allowed for the downstream joint of Link *id* (m/s for prismatic joint, rad/s for revolute joints).
$OutputFolder	Folder for output files.

Table 4: *The parameters used to initialize a boom crane object.*

Sample Boomcrane Init File

Here is the contents of a sample init file:

```
$VERBOSE 0

//Link 1
$0|LinkDHParameters 90 -0.65 1.4 0
$0|LinkJointVariable Theta

$0|LinkJointVelocity 0

$0|LinkCGLocation -0.35 0.7 0
$0|LinkCGOrientation 0 0 0

$0|LinkMass 682.97 682.97 682.97
$0|LinkMomentOfIntertia 139.44 139.44 55.776

$0|JointControlType position

$0|JointProportionalCoefficient 1000000000
$0|JointIntegralCoefficient 1
$0|JointDerivativeCoefficient 100000000

$0|MaximumActuatablePower 500000

$0|UpstreamJointLimits -1e30 1e30 //no limits
$0|JointLimitStiffness 1000000
$0|JointLimitDamping 10000

// Link 2

$1|LinkDHParameters 0 1 0 45
$1|LinkJointVariable Theta

$1|LinkJointVelocity 0

$1|LinkCGLocation 0.5 0 0
$1|LinkCGOrientation 0 0 0

$1|LinkMass 754 754 754
```

```
$1|LinkMomentOfIntertia 31.42 78.5 78.5

$1|JointControlType position

$1|JointProportionalCoefficient 1000000000
$1|JointIntegralCoefficient 1
$1|JointDerivativeCoefficient 100000000

$1|MaximumActuatablePower 500000

$1|UpstreamJointLimits 0 90
$1|JointLimitStiffness 10000000
$1|JointLimitDamping 1000000

$OutputFolder boomcrane_output_folder
```

3.2.6 Winch Initialization File Specification

The following table provides a description of each entry of an `SMS::Winch .ini` file:

Variable Name	**Variable Description**
$Kd	Derivative gain
$Kp	Proportional gain
$Ki	Integral gain
$accel_max	Maximum acceleration
$decel_max	Maximum deceleration (negative)
$VelocitySetPoint	Velocity set point (m/s)
$TensionSetPoint	Tension set point (N)
$OutputFolder	A folder where output files will be created and stored.

Table 5: *Winch Class init file description*

Sample Winch Init File

Here is the contents of a sample init file:

```
$VelocitySetPoint 0
$TensionSetPoint 2000

$accel_max 3
$decel_max -3

$Kp 0.1
$Kd 0.01
$Ki 0.0001

$OutputFolder winch_output_folder
```

3.3 Class Methods

This section provides descriptions of the important methods used to interact with the simulation objects. Some of the methods are common to all simulation objects since they were inherited from `SMS::SimObject` abstract class. For examples on how these methods are used within the context of a computer program, please refer to the SMS Validation Test Suite.

3.3.1 Cable Class

3.3.1.1 Creating a Cable Object

An `SMS::Cable` object can be generated using the `SMS::Cable` class constructor:

```
SMS::Cable myCable(std::string cable_init_file,
double t0,
bool file_output);
```

where `cable_init_file` is a string that contains the location of the cable's initialization file (defined in Section 3.2.3), `t0` is the start time of the cable object and `file_output` is set to *true* to allow simulation data to be output to disk.

3.3.1.2 Controlling the Position and Velocity of End Nodes

The SMS::Cable object may be fixed to other objects, such as a ship. Attached objects kinematically constrain the position of their connected end nodes. The position and velocity of the end nodes of the cable may be specified using the following function:

```
myCable.SetNode0BoundaryConditions(ub::vector<double> Node0Pos,
 ub::vector<double> Node0Vel,
 ub::vector<double> Node0Accel);
```

or

```
myCable.SetNodeNBoundaryConditions(ub::vector<double> NodeNPos,
ub::vector<double> NodeNVel,
ub::vector<double> NodeNAccel);
```

where Node0Pos, NodeNPos, Node0Vel, and NodeNVel, are vectors of length 3 containing the 3D position and velocity of the node to be controlled. It should be noted that discontinuous positions and velocities with respect to time will cause cable model simulation instabilities. The SMS::Cable end node position, velocity, or acceleration can also each be modified individually using one of the following functions:

```
myCable.SetNode0Pos(ub::vector<double> Node0Pos);
myCable.SetNode0Vel(ub::vector<double> Node0Vel);
myCable.SetNode0Accel(ub::vector<double> Node0Accel);
myCable.SetNodeNPos(ub::vector<double> NodeNPos);
myCable.SetNodeNVel(ub::vector<double> NodeNVel);
myCable.SetNodeNAccel(ub::vector<double> NodeNAccel);
```

Take note that when SMS::SimObjects enter an SMS API master-slave relationship, boundary conditions are automatically updated by the SMS API; thus, calling these methods is not required. In such a case, these calls aren't forbidden, though they will have no effect since the boundary conditions are automatically updated by the master during the AdvanceTime() call.

3.3.1.3 Adding a Winch to a Cable

An SMS::Winch object can be added to an end node of a cable. The SMS::Winch object, which has it's own control methods, will control the length of the element it is connected

to. If the element is too long, a node is inserted to the `SMS::Cable` to prevent cable elements becoming undesirably long.

An `SMS::Winch` is added to node 0 or node N of a cable using the following functions:

```
myCable.AddWinch0(SMS::Winch &winch);  \\Attach Winch to Node 0
```

and

```
myCable.AddWinchN(SMS::Winch &winch);  \\Attach Winch to Node N
```

where `winch` is a reference to the SMS::Winch object being attached.

3.3.1.4 Detaching an Attached Winch Object

An attached `SMS::Winch` object is removed from a cable using the following function:

```
myCable.DetachWinch(Winch *winch);
```

where `winch` is a pointer to the `SMS::Winch` object being detached.

3.3.1.5 Getting Loads a Cable is Applying to Attached Objects

The loads an `SMS::Cable` is applying to an attached `SMS::Boomcrane` or `SMS::Payload` can be obtained, depending on which node the `SMS::SimObject` is attached to, using:

```
myCable.GetNode0Load(ub::vector<double> cableForce);
```

or

```
myCable.GetNodeNLoad(ub::vector<double> cableForce);
```

where `cableForce` is a 6×1 spatial vector containing both the force and moment components.

3.3.1.6 Getting the Position of a Point Along a Cable

To obtain the Cartesian position of a point along an `SMS::Cable`, the user will call:

```
myCable.GetPosition(double sPos,
ub::vector<double> &pos);
```

where `sPos` is the arclength distance along a cable, and `pos` is a 3×1 Cartesian position vector returned by the function.

3.3.1.7 Getting the Velocity of a Point Along a Cable

To obtain the Cartesian velocity of a point along an `SMS::Cable`, the user will call:

```
myCable.GetVelocity(double sPos,
ub::vector<double> &vel);
```

where `sPos` is arclenth distance along a cable, and `vel` is a 3×1 Cartesian velocity vector returned by the function.

3.3.1.8 Getting the Acceleration of a Point Along a Cable

To obtain the Cartesian acceleration of a point along an `SMS::Cable`, the user will call:

```
myCable.GetAcceleration(double sPos,
ub::vector<double> &accel);
```

where `sPos` is arclength distance along a cable, and `accel` is a 3×1 Cartesian acceleration vector returned by the function.

3.3.1.9 Getting the Cable Tangent at a point Along a Cable

To obtain the tangent as a vector at a point along an `SMS::Cable`, the user will call:

```
myCable.GetTangent(double sPos,
ub::vector<double> &tangent);
```

where `sPos` is arclength distance along a cable, and `tangent` is a 3×1 Cartesian tangent vector returned by the function.

3.3.1.10 Applying External Forces to a Cable

To apply a force at a point along an `SMS::Cable`, the user will call:

```
myCable.AddForce(double sPos,
ub::vector<double> &force);
```

where `sPos` is arclength distance along a cable the force will be applied to, and `force` is a 3×1 Cartesian force vector being applied to the `SMS::Cable`.

3.3.1.11 Extracting State Information

To get the most recent state vector from an `SMS::Cable` object, defined in section 3.4.1, the `GetState` function is used:

```
myCable.GetState(std::vector<double> &X);
```

where a vector `X` is passed to the function so that it will be populated with the state vector of the `SMS::Cable`. The format of the state vector is described in Section 3.4.1

3.3.1.12 Manually Setting the Cable Boundary Conditions

The boundary conditions of an `SMS::Cable` can be set using:

```
myCable.SetNode0BoundaryConditions(std::vector<double> Node0Pos,
std::vector<double> Node0Vel,
std::vector<double> Node0drds,
std::vector<double> Node0P1,
std::vector<double> Node0P2);
```

and

```
myCable.SetNodeNBoundaryConditions(std::vector<double> NodeNPos,
std::vector<double> NodeNVel,
std::vector<double> NodeNdrds,
std::vector<double> NodeNP1,
std::vector<double> NodeNP2);
```

where `Node0Pos` and `NodeNPos` are 3×1 vectors containing the Cartesian position of nodes 0 and N respectively, `Node0Vel` and `NodeNVel` are 3×1 vectors containing the Cartesian velocities of nodes 0 and N respectively, `Node0drds` and `NodeNdrds` are 3×1 vectors containing the Cartesian tangent vector of nodes 0 and N respectively, `Node0P1` and `NodeNP1` are 3×1 vectors containing the twisted Frenet frame axis $P1$ unit vector of nodes 0 and N respectively, and `Node0P2` and `NodeNP2` are 3×1 vectors containing the twisted Frenet frame axis $P2$ unit vector of nodes 0 and N respectively.

3.3.1.13 Advancing the Cable Through Time

To advance the cable dynamics through time, to go from one state to a future state `stepSize` seconds in the future, the `AdvanceTime` function is called:

```
myCable.AdvanceTime(double stepSize);
```

After `AdvanceTime` is called the latest state information can be retrieved.

3.3.1.14 Handling Added Mass

The hydrodynamic effect of added mass for `SMS::Cable`s is handled by adding discretised mass at locations along a cable:

```
myCable.AddMass(double sPos,
double mass);
```

where `sPos` is the arclength distance along a cable, and `mass` is the added mass being added to the cable at `sPos`.

The `AddMass()` function is cumulative and must be manually cleared to zero the accumulator. This can be accomplished by calling:

```
myCable.ClearMasses();
```

Take note that the SMS API user is responsible to ensure a complete hydrodynamics model is used, this could include drag (a function of relative velocity to the fluid), added mass, and fluid inertial loading (a function of absolute acceleration of the fluid).

3.3.1.15 Outputting Cable State Data to Disk

Between `AdvanceTime` calls, the current State of a cable can be output (appended) to the file specified in the `SMS::Cable` ini file. This is accomplished by calling the function:

```
myCable.OutputSimObjectData();
```

The format of the output files for an `SMS::Cable` are described in Section 3.5.1.

3.3.2 Winch Class

3.3.2.1 Creating a Winch Object

The winch class constructor definition is:

```
 myWinch(std::string init_file,
double t0,
bool file_output);
```

where `init_file` is a string containing the name of the ASCII text file that defines the winch's initialization parameters. The time offset to start the simulation at is specified by `t0` and `file_output` is set to *true* to allow simulation data to be output to disk.

3.3.2.2 Setting Winch Control Mode

The `SMS::Winch` control mode can be modified using:

```
myWinch.SetMode(std::string mode);
```

where `mode` is either "velocity" or "tension".

3.3.2.3 Setting Winch Velocity Control Setpoint

The velocity control setpoint can be modified using:

```
myWinch.SetVelocitySetPoint(double setpoint);
```

where `setpoint` is the payout or payin speed in m/s, negative values mean payin while positive values mean payout.

3.3.2.4 Setting Winch Tension Control Setpoint

The tension control setpoint can be modified using:

```
myWinch.SetTensionSetPoint(double setpoint);
```

where `setpoint` is the tension setpoint in Newtons.

3.3.2.5 Setting the acceleration limits

The acceleration limits of the winch can be modified using:

```
myWinch.SetAccelDecelLimits(double accel,
double decel);
```

where `accel` is positive and `decel` is negative.

3.3.3 Payload Class

3.3.3.1 Creating a Payload Object

An `SMS::Payload` object can be generated using the Cable class constructor:

```
SMS::Payload myPayload(std::string payload_init_file,
double t0,
bool file_output);
```

where `payload_init_file` is a string that contains the location of the payload's initialization file (defined in Section 3.2.4), `t0` is the start time of the payload object, and `file_output` is set to *true* to allow simulation data to be output to disk.

3.3.3.2 Setting a Payload's Position and Orientation Manually

The position and orientation of an `SMS::Payload` can be set using:

```
myPayload.SetPosOri(ub::vector<double> posOri,
ub::vector<double> posOriDot);
```

where `posOri` is a 6×1 array containing the payload's 3 Cartesian position components relative to the Earth-fixed frame followed by the 3 Z-Y′-X″ Euler orientation components, and `posOriDot` the time derivative of `posOri`.

3.3.3.3 Attaching Cables

An `SMS::Cable` can be attached to an `SMS::Payload` using either a pinned (spherical joint) or clamped connection using:

```
myPayload.AddCableClamped(SMS::Cable *cable_to_attach,
ub::vector<double> &location,
ub::vector<double> &p1,
ub::vector<double> &p2,
int end);
```

or

```
myPayload.AddCablePinned(SMS::Cable *cable_to_attach,
ub::vector<double> &location,
int end);
```

where `cable_to_attach` is a pointer to the cable being attached, `location` is a 3×1 Cartesian vector of the cable attachment location relative to the `SMS::Payload`'s body-fixed frame, `p1` and `p2` are two perpendicular vectors, expressed with respect to the body-fixed frame whose cross-product will define the cable tangent at its attachment end, and `end` is set to 0 to attach the node 0 to the `SMS::Payload` or 1 to attach node N.

3.3.3.4 Detaching a Cable

A previously attached cable can be removed from the payload using:

```
myPayload.DetachCable(Cable *cable_to_detach);
```

where `cable_to_detach` is a pointer to an attached cable. It's a good idea to reduce the tension in the cable to 0 before detaching a cable.

3.3.3.5 Manually Applying External Forces

External forces, such as hydrodynamics forces, can be added to the payload using

```
myPayload.ApplyExternalForce(ub::vector<double> &force);
```

where `force` is a 6×1 spatial vector containing the force and moment being applied to the `SMS::Payload` expressed with respect to the Earth-fixed frame. Adding forces via this method is cumulative,and the force summation must be manually cleared. This is accomplished using:

```
myPayload.ClearForces();
```

3.3.3.6 Applying Added Mass

The hydrodynamic effect of added mass for an `SMS::Payload` payload object can be handled by adding

```
myPayload.AddAddedMass(ub::matrix<double> &mass);
```

where `mass` is the added mass being added to the `SMS::Payload`.

The `AddAddedMass()` function is cumulative and must be manually cleared to zero the accumulator. This can be accomplished by calling:

```
myPayload.ClearAddedMass();
```

3.3.3.7 Getting the State of the Payload

The state of an `SMS::Payload` can be obtained using:

```
myPayload.GetState(ub::vector<double> &X);
```

where `X` is a 12×1 matrix containing the state vector of the `SMS::Payload` described in 3.4.2

3.3.3.8 Getting the Payload Position

To obtain the position and orientation information of an `SMS::Payload`, the user will call:

```
myPayload.GetPosition(ub::vector<double> &pos_ori);
```

where `pos_ori` is a 6×1 array that will be filled with the Cartesian position vector and the 3 Euler angle components describing orientation.

3.3.3.9 Getting the Payload Velocity

To obtain the 6 DOF velocity vector of an `SMS::Payload`, the user will call:

```
myPayload.GetVelocity(ub::vector<double> &vel);
```

where `vel` is a 6×1 spatial vector that will be filled with the Cartesian linear and angular velocity vector components respectively.

3.3.3.10 Getting the Payload Acceleration

To obtain the 6 DOF acceleration vector of an `SMS::Payload`, the user will call:

```
myPayload.GetAcceleration(ub::vector<double> &accel);
```

where `accel` is a 6×1 spatial vector that will be filled with the Cartesian linear and angular acceleration vector components respectively.

3.3.3.11 Getting Boundary Condition Information for Attached Cables

If an `SMS::Cable` object is attached to an `SMS::Payload` object, the following function can be called to aquire the position and velocity of the cable connection point:

```
myPayload.GetBCInfo(unsigned int cable_id,
ub::vector<double> &vel,
ub::vector<double> &pos);
```

where `cable_id` is the attached cable's, and `pos` and `vel` are the 3×1 Cartesian position and velocity vectors of the cable attachment point.

3.3.3.12 Advancing the Payload Through Time

To go from one state to some state `stepSize` seconds in the future, the `AdvanceTime` function is called:

```
myPayload.AdvanceTime(double stepSize);
```

After `AdvanceTime` completes its run, the latest state information can be retrieved from the `SMS::Payload`. If `myPayload` is the master in a set of attached simulation objects, the `AdvanceTime` call will also advance the time of all slave objects.

3.3.3.13 Outputting Payload State Data to Disk

Between `AdvanceTime` calls, the current state of an `SMS::Payload` can be written (appended) to the file specified in the `SMS::Payload` ini file. This is accomplished by calling the function:

```
myPayload.OutputSimObjectData();
```

The format of the output files for an `SMS::Payload` are described in Section 3.5.2.

3.3.4 BoomCrane Class

3.3.4.1 Creating a Boomcrane Object

An `SMS::Boomcrane` object can be generated using the `SMS::Boomcrane` class constructor:

```
SMS::BoomCrane myBoomCrane(std::string boomcrane_init_file,
double t0,
bool file_output);
```

where `boomcrane_init_file` is a string that contains the location of the cable's initialization file (defined in Section 3.2.3), `t0` is the start time of the cable object and `file_output` is set to *true* to allow simulation data to be output to disk.

3.3.4.2 Setting the Boom Crane's Base Position and Orientation

The `SMS::Boomcrane` base position is set using:

```
myBoomcrane.SetBaseGlobalPositionAndOrientation(
ub::vector<double> &posOri,
ub::vector<double> &velocity,
ub::vector<double> &acceleration);
```

where `posOri` is a 6×1 array holding the 3 Cartesian position components and 3 Z-Y′-X″ Euler angles components that describe the orientation, `velocity` is a 6×1 spatial vector holding the velocity and angular velocity, and `acceleration` is a 6×1 spatial vector holding the linear and angular acceleration.

3.3.4.3 Actuating the Boom Crane Joints

Each joint of an `SMS::Boomcrane` is controlled by setting a joint controller target velocity or displacement:

```
myBoomcrane.SetTargetJointVelocity(int jointID,
double targetVelocity);
```

or

```
myBoomcrane.SetTargetJointDisplacement{int JointID,
double targetDisplacement};
```

where `jointID` is the *id* of the joint (starting at 0), `targetDisplacement` is the displacement the user wishes the joint to achieve, and `targetVelocity` is the velocity the user wishes the joint to achieve.

3.3.4.4 Adding a Cable to the Boom Crane

Automatic Boundary Condition Synchronisation

An `SMS::Cable` can be attached to an `SMS::Payload` using either a pinned (spherical joint) or clamped connection using:

```
myBoomcrane.AddCablePinned(int linkID,
SMS::Cable *cable_to_attach,
ub::vector<double> location,
int end)
```

or

```
myBoomcrane.AddCableClamped(int linkID,
SMS::Cable *cable_to_attach,
ub::vector<double> location,
ub::vector<double> p1,
ub::vector<double> p2,
int end);
```

where `cable_to_attach` is a pointer to a pre-initialised cable, `linkID` is the *id* of the `SMS::Boomcrane` link the cable is connecting to, `location` is a vector that defines the connection point described with respect to the link's rigid body's body-fixed frame, `end` is 0 to attach node 0 of the cable and 1 to attach node N. For the case of clamped connections, the orthogonal vectors `p1` and `p2` define a frame fixed to the boomcrane link where $p1 \times p2$ will define the cable's tangent at the connection point (fixed relative to the rigid body).

Attaching an `SMS::Cable` with these functions allows the SMS API to assign master-slave relationships between the `SMS::SimObjects`. This means the boundary conditions are automatically synchronised and the `SMS::Cable`, if not a slave already, will become

a slave and have its dynamics automatically evolved by the master SMS::SimObject. If cable_to_attach already has a master, then myBoomcrane will become a slave to cable_to_attach, and cable_to_attach's master will handle the evolution of the dynamics through time for all slaves. If cable_to_attach is not yet a slave, then myBoomcrane will become its master.

Manual Boundary Condition Synchronisation

An SMS::Cable can be attached to an SMS::Payload using either a pinned (spherical joint) or clamped connection using:

```
myBoomcrane.AddCablePinned(int linkID,
ub::vector<double> location,
int end)
```

or

```
myBoomcrane.AddCableClamped(int linkID,
ub::vector<double> location,
ub::vector<double> p1,
ub::vector<double> p2,
int end);
```

where linkID is the *id* of the SMS::Boomcrane link the cable is connecting to, location is a vector that defines the connection point described with respect to the link's rigid body's body-fixed frame, end is 0 to attach node 0 of the cable and 1 to attach node N. For the case of clamped connections, the orthogonal vectors p1 and p2 define a frame fixed to the boomcrane link where $p1 \times p2$ will define the cable's tangent at the connection point (fixed relative to the rigid body).

The boundary conditions of the connection between the SMS::Cable and myBoomcrane will have to be manually updated, and the dynamics both be individually evolved through time. Take note that manual updates of clamped boundary conditions are not yet fully supported.

3.3.4.5 Getting Boundary Condition Information for Attached Cables

The SMS API user is currently limited to acquiring the boundary conditions for pinned cables only. This can be done by calling:

```
myPayload.GetBCInfo(unsigned int joint_id,
unsigned int cable_id,
ub::vector<double> &vel,
ub::vector<double> &pos);
```

where `joint_id` is the id of the proximal joint of the `SMS::ABA_RigidBody` the cable is attached to, `cable_id` is the attached cable id (only one cable can currently be attached per `SMS::ABA_RigidBody`), `pos` and `vel` are the 3×1 Cartesian position and velocity vectors of the cable attachment point.

3.3.4.6 Detaching a Cable from the Boom Crane

A currently attached `SMS::Cable` can be removed from the `SMS::Boomcrane` using:

```
myBoomcrane.DetachCable(Cable *cable_to_detach);
```

where `cable_to_detach` is a pointer to an attached cable.

3.3.4.7 Applying External Forces

External forces are applied to an `SMS::Boomcrane` using:

```
myBoomcrane.ApplyExternalForce(int linkID,
std::vector<double> forceVector);
```

where `linkID` is the *id* of the link in the boom crane the contact force is being applied to, `forceVector` is a length 3 vector of the force.

These external forces are cumulative. It may be necessary to clear them before applying new forces using:

```
myBoomcrane.ClearRigidBodyForces();
```

3.3.4.8 Extracting State Information

To get the most recent state vector from a boom crane object, the `GetState` function is used:

```
 myBoomcrane.GetState(std::vector<double> &X);
```

where `X` is a vector that will be populated with the state information. The format of the state vector is described in Section 3.4.3.

3.3.4.9 Advancing the Boom Crane Through Time

The `SMS::Boomcrane` is advanced through time using:

```
myBoomcrane.AdvanceTime(double stepSize);
```

where `stepSize` is the amount of time to evolve the `SMS::Boomcrane` dynamics over.

3.3.4.10 Outputting Boom Crane State Data to Disk

Between `AdvanceTime` calls, the current State of an `SMS::Boomcrane` can be written (appended) to the file specified in the `SMS::Boomcrane` ini file. This is accomplished by calling the function:

```
myBoomcrane.OutputSimObjectData();
```

The format of the output files for an `SMS::Boomcrane` are described in Section 3.5.3.

3.3.5 Contact Resolution

After an `SMS::Payload` object is initialised in code, the user can make use of the collision resolution system to resolve collisions between the `SMS::Payload` and an external object.

3.3.5.1 Creating a Collision Object for the Payload

To provide the `SMS::Payload` with its collision geometry, primitives can be added to construct the shape using:

```
myPayload.AddPrimitiveContactPolyhedron(std::string primitive_type,
double length,
double width,
double height,
int numLengthSegments,
int numWidthSegments,
int numHeightSegments,
ub::vector<double> &posOri);
```

or user-defined closed, convex `.3ds` objects can be added to construct the shape using

```
myPayload.Add3DSContactPolyhedron(std::string file_name,
double scaleX,
double scaleY,
double scaleZ,
ub::vector<double> &posOri);
```

where `primitive_type` is a string containing the type of primitive to create set to "box", "cylinder", or "sphere" [4], `length`, `width`, and `height` are the length, width and height dimensions of the primitive being created, `numLengthSegments`, `numWidthSegments`, and `numHeightSegments` are the number of segments to subdivide the faces of the primitive along that dimension (this refines the polyhedral mesh), `file_name` is the name of the `.3ds` file containing a convex closed polyhedron, `scaleX`, `scaleY` and `scaleZ` is the amount to scale the `.3ds` polyhedron by (generally set to 1), and `posOri` is a 6×1 array containing the 3 Cartesian position vector components and 3 Euler angle rotations to describe the orientation with respect to the `SMS::Payload`'s body-fixed frame.

[4] Only box primitives are currently implemented in the SMS API.

3.3.5.2 Creating an External Collision Object

To provide the `SMS::Payload` with external collision geometry for it to collide with, primitives can be added to construct the shape using:

```
myPayload.AddExternalPrimitiveContactPolyhedron(
std::string primitive_type,
double length,
double width,
double height,
int numLengthSegments,
int numWidthSegments,
int numHeightSegments,
ub::vector<double> &posOri));
```

or user-defined closed, convex `.3ds` objects can be added to construct the shape using:

```
AddExternal3DSContactPolyhedron(std::string file_name,
double scaleX,
double scaleY,
double scaleZ,
ub::vector<double> &posOri);
```

where `primitive_type` is a string containing the type of primitive to create set to "box", "cylinder", or "sphere" [5], `length`, `width`, and `height` are the length, width and height dimensions of the primitive being created, `numLengthSegments`, `numWidthSegments`, and `numHeightSegments` are the number of segments to subdivide the faces of the primitive along that dimension (this refines the polyhedral mesh), `file_name` is the name of the `.3ds` file containing a convex closed polyhedron, `scaleX`, `scaleY` and `scaleZ` is the amount to scale the `.3ds` polyhedron by (generally set to 1), and `posOri` is a 6×1 array containing the 3 Cartesian position vector components and 3 Euler angle rotations to describe the orientation relative to the origin of the external object.

3.3.5.3 Setting Payload and External Object's Material Properties for Contact

The material properties of the `SMS::Payload`'s material contact properties can be manually set using:

[5] Only box primitives are currently implemented in the SMS API.

```
myPayload.SetMaterialProperties(double K,
double B);
```

and similarly for external object.

```
myPayload.SetExternalObjectMaterialProperties(double K_ext,
double B_ext);
```

where `K` and `K_ext` are the material stiffness coefficients (Young's modulus), and `B` and `B_ext` are the material damping coefficients

3.3.5.4 Setting the Position, Velocity and Acceleration of the External Collision Object

The position, velocity and acceleration of the external collision object's origin relative to the Earth-fixed frame can be defined using:

```
myPayload.SetExternalContactPolyhedraPositionOrientation(
ub::vector<double> &posOri,
ub::vector<double> &posOriDot,
ub::vector<double> &posOriDotDot);
```

where `posOri` is a 6×1 array containing the 3 Cartesian position vector components, and 3 Euler angle rotations to describe the orientation, and `posOriDot` and `posOriDotDot` are the first and second time derivative of `posOri`.

3.3.5.5 Outputting Payload Contact Resolution State Data to Disk

Between `AdvanceTime` calls, information about the `SMS::Payload` contact resolution system are output to file along with the `SMS::Payload`'s files during the `SMS::Payload`'s call of:

```
myPayload.OutputSimObjectData();
```

The latter two are only given if the contact resolution system is employed, and their contents are discussed in 3.5.5.

3.4 Class State Vector Definitions

3.4.1 Cable State Vector Definition

```
Node 0 X Velocity (m/s)
Node 0 X Position (m)
Node 0 Y Velocity (m/s)
Node 0 Y Position (m)
Node 0 Z Velocity (m/s)
Node 0 Z Position (m)
Node 1 X Velocity (m/s)
Node 1 X Position (m)
Node 1 Y Velocity (m/s)
Node 1 Y Position (m)
Node 1 Z Velocity (m/s)
Node 1 Z Position (m)
.
.
.
Node N-1 X Velocity (m/s)
Node N-1 X Position (m)
Node N-1 Y Velocity (m/s)
Node N-1 Y Position (m)
Node N-1 Z Velocity (m/s)
Node N-1 Z Position (m)
Node N X Velocity (m/s)
Node N X Position (m)
Node N Y Velocity (m/s)
Node N Y Position (m)
Node N Z Velocity (m/s)
Node N Z Position (m)
Element 1 Length (m)
Element N Length (m)
```

3.4.2 Payload State Vector Definition

```
X velocity (m/s)
Y velocity (m/s)
Z velocity (m/s)
```

```
phi   dot angular velocity (rad/s)
theta dot angular velocity (rad/s)
psi   dot angular velocity (rad/s)
X position (m)
Y position (m)
Z position (m)
phi   rotation (rad)
theta rotation (rad)
psi   rotation (rad)
```

3.4.3 Boomcrane State Vector Definition

```
X velocity (m/s)
Y velocity (m/s)
Z velocity (m/s)
phi   dot angular velocity (rad/s)
theta dot angular velocity (rad/s)
psi   dot angular velocity (rad/s)
Joint 1 velocity
Joint 2 velocity
.
.
.
Joint i velocity
X position (m)
Y position (m)
Z position (m)
phi   rotation (rad)
theta rotation (rad)
psi   rotation (rad)
Joint 1 displacement
Joint 2 displacement
.
.
.
Joint i displacement
```

3.4.4 Winch State Vector Definition

This winch implementation has no state since it currently does not use its integrator. Instead, the `SMS::Cable` uses attached `SMS::Winch` objects to obtain the payout rate. The payout rate then is used as the time derivative of the `SMS::Cable`'s attached element length which gets integrated by the `SMS::Cable`'s integrator.

3.5 Class Output File Formats

3.5.1 Format of Cable Output Files

The output files for an `SMS::Cable` object are `results.dat`, `tensions.dat`, and `curvatures.dat`. Each row of these output files represents the output for an instant in time. The first column of each file supplies the time stamp for the instant in time. The remaining columns are different depending on the output file.

3.5.1.1 results.dat

Each row of the `results.dat` file contains the time stamp and the state vector (see Section 3.4.1) of the cable for each instant in time.

3.5.1.2 tensions.dat

Each row of the `tensions.dat` file contains the time stamp and the tension in each element. The $(i+1)^{th}$ column contains the tension in element i.

3.5.1.3 curvatures.dat

Each row of the `curvatures.dat` file contains the time stamp and each 3 consecutive column components hold the 3 Cartesian curvature vector components, $\mathbf{r}''$, in Earth-fixed coordinates, at each cable node starting with node 0.

3.5.2 Format of Payload Output Files

The output files for an `SMS::Payload` object are `results.dat`, `cable_forces.dat`, `contact_forces.dat` and `contact_geometry.dat`. The latter two are only written if the contact resolution system is employed. Their contents are discussed in Section 3.5.5. Each row of these output files holds simulation information for the `SMS::Payload` for an instant in time. The first column supplies the time stamp for the instant in time. The remaining columns are different depending on the output file.

3.5.2.1 results.dat

Each row of the `results.dat` file contains the time stamp and the state vector (see Section 3.4.2) of the payload for each instant in time.

3.5.2.2 cable_forces.dat

Each row of the `cable_forces.dat` file contains the time stamp and the 6×1 spatial vector of the force containing the linear force and moment applied by the `SMS::Cable` at

the connection point. There is currently only support for a single cable to be attached at a time.

3.5.3 Format of Boomcrane Output Files

The output files for an `SMS::Boomcrane` object consist of only `results.dat`. Each row of the output files holds simulation information for the `SMS::Boomcrane` for an instant in time. Each row of the file contains the time stamp and the state vector (see Section 3.4.3) of the `SMS::Boomcrane` for each instant in time.

3.5.4 Format of Winch Output Files

The only output file for an `SMS::Winch` object is `winch_velocity.dat`. Each row of this output file represents the output for an instant in time. The first column of the file supplies the time stamp for that instant in time. The other column holds the payout/payin velocity of the cable at that time.

3.5.5 Format of Payload Contact Resolution Output Files

The output files for an `SMS::Payload` object, if contact resolution is used, consist of `contact_forces.dat` and `contact_geometry.dat`. Each row of these output files holds simulation information for the contact resolution system of the `SMS::Payload` for an instant in time. The first column of each file supplies the time stamp for the instant in time. The remaining columns are different depending on the output file.

3.5.5.1 contact_forces.dat

Each row of the `contact_forces.dat` file contains the time stamp, the three contact force vector components (in Earth-fixed coordinates), the three contact force location components (in Earth-fixed coordinates), stiffness component of the force and the damping component of the force.

3.5.5.2 contact_geometry.dat

Each row of the `contact_geometry.dat` file contains the time stamp, the volume of interference for the contact and the penetration depth of the contact.

4 SMS API Validation

An SMS API Validation Test Suite, consisting of a set of test cases used to verify the fidelity of the SMS API based simulations, has been implemented and submitted as part of this report. Each of the tests is described in the following section, including the motivation behind each test, the experimental setup as well as the results of simulation. The SMS Validation Test Suite should also be referred to for examples of how the SMS API is used to set up and execute similar simulations.

4.1 Cable Class Validation

4.1.1 Beam Deflection Test

4.1.1.1 Motivation

The purpose of this test is to validate the cable bending stresses by comparing against an analytical solution.

4.1.1.2 Experimental Setup

The deflection δ of a cantilevered beam of length $L = 3$ m due to a tip load of $F = 5e4$ N as shown in Figure 28 is calculated as:

$$\delta = \frac{FL^3}{3EI} \tag{105}$$

where $EI = 1e6$ Nm^2 is the bending stiffness of the cable. A simulation was executed, in a gravity free environment, by modeling the beam with an `SMS::Cable` object made of 3 elements of equal lengths. The expected cable deflection of 0.45 m, was compared against the simulation results.

In addition, the period of oscillation τ_{beam} of the cantilevered beam is calculated as [42]:

$$\tau_{beam} = \frac{\pi}{3.52\sqrt{\frac{EI}{wL^4}}} \tag{106}$$

where w is the weight per unit length of the cable and can be calculated using the `SMS::Cable`'s diameter of 0.1 m and its material density of 1000 kg/m^3. The expected period of oscillation of the beam, 0.045022 s, was compared against the simulation results.

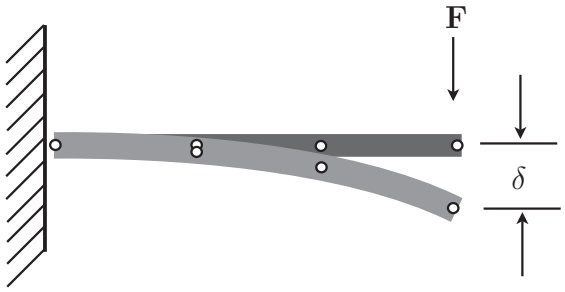


Figure 28: *Schematic of the beam bending test that will be used to validate the cable model*

4.1.1.3 Simulation Results

Figure 29 shows the deflection of the cantilever beam tip over time. The period of oscillation was found to be 0.046612 seconds, a 3.5% difference of predicted. The average tip deflection was found to be 0.434 m, a 3.5% difference from predicted. Take note that some light hydrodynamic forces were supplied to the cable model to provide some vibrational damping and help maintain simulation stability.

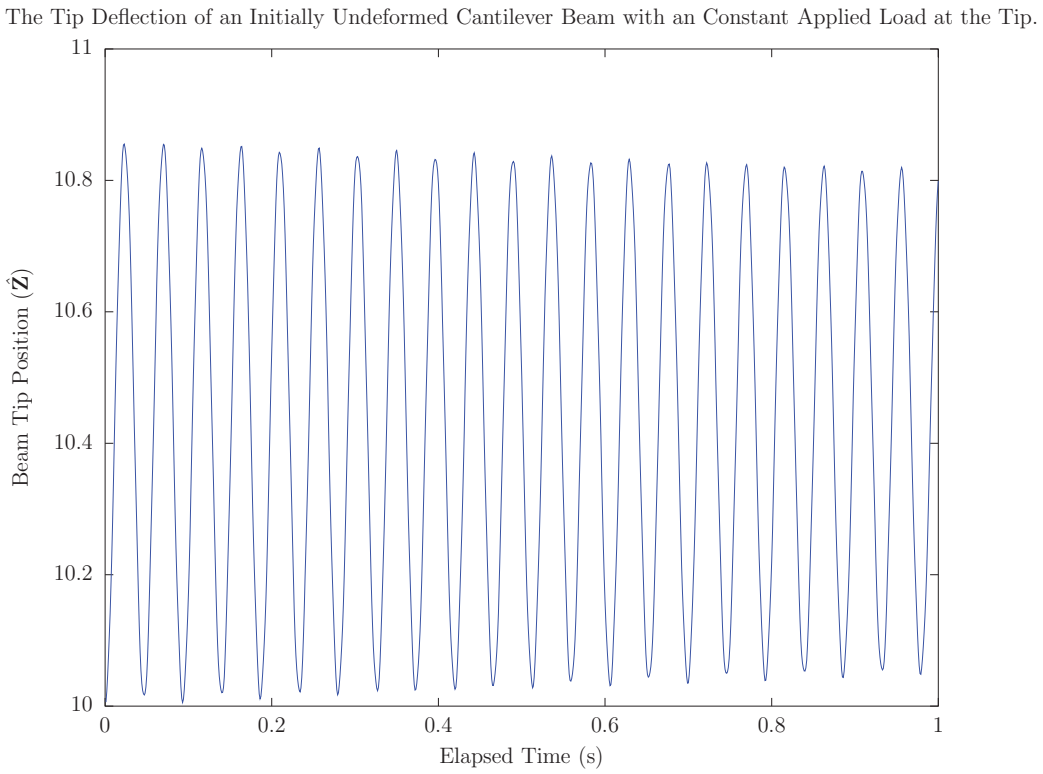


Figure 29: *The tip deflection of vibrating beam modelled with an* `SMS::Cable`.

4.1.2 Pendulum Period of Oscillation Test

This test has been implemented in tandem with a Payload validation test and can be found in Section 4.3.5.

4.1.3 Torsion Period of Oscillation Test

This test has been implemented in tandem with a Payload validation test and can be found in Section 4.3.6.

4.1.4 Top Node Oscillation Test

4.1.4.1 Motivation

The purpose of this test is to demonstrate the SMS API's capability to simulate both slack and taut cables.

4.1.4.2 Experimental Setup

To qualitatively show the capabilities of the `SMS::Cable` model in low tension and high tension states, within a single simulation, a 20 m long verticle cable with 4 elements of equal lengths is simulated with a `SMS::Payload` attached at the bottom node while the top node of the `SMS::Cable` is oscillated in three dimensions as shown in Figure 30. The free end of the cable will be oscillated in three dimensions according to the following equations of motion:

$$\mathbf{r}^{(0)} = [A_x \sin(\omega_x t + \phi_x) \quad A_y \sin(\omega_y t + \phi_y) \quad A_z \sin(\omega_z t + \phi_z)]^T \tag{107}$$

where $A_x, A_y, A_z = 2, 2, 2$ are the amplitudes of oscillation in the X, Y, and Z directions respectively, $\omega_x, \omega_y, \omega_z = 0.09, 0.05, 1.5$ are the frequency of oscillation in the X, Y, and Z directions respectively, $\phi_x, \phi_y, \phi_z = 0, 0, 0$ are the phase angle offsets of oscillation in the X, Y, and Z directions respectively and t is time.

The `SMS::Cable` has a bending stiffness of 0.0 Nm^2, axial stiffness of $8.0e5$ N, and torsional stiffness of 0.0 Nm^2, a diameter of 0.005 m a density of 7700 kg/m^3 and a `CID` set at $5e3$. Node N of the cable is attached to the `SMS::Payload` at its body-fixed frame origin. The `SMS::Payload` is modeled as a sphere with a radius of 0.7 m, a mass of 1400 kg, a density of 1000 kg/m^3 and is located at $[0, 0, -20]^T$.

The simulated tensions in the `SMS::Cable` as well as the position of its Node N, the same as the `SMS::Payload`'s body-fixed frame origin, are plotted over time in Section 4.1.4.3.

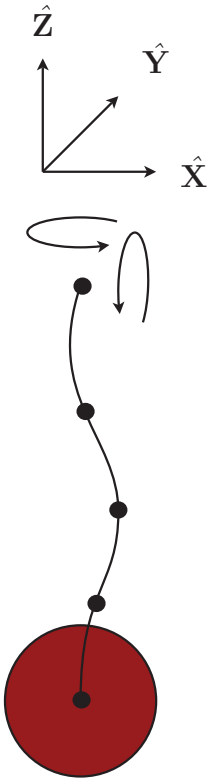


Figure 30: *The figure shows the test that will demonstrate both the low tension and high tension capabilities that the chosen cable model is capable of modeling. The top node of the cable will be oscillated in three dimensions. The resulting harmonic motion of the free end of the cable is shown.*

4.1.4.3 Simulation Results

As anticipated, Figure 31 show the payload is heavily influenced by large $\hat{\mathbf{Z}}$ oscillations of the top node, while the period of oscillation of the pendulum is low enough that the $\hat{\mathbf{X}}$ and $\hat{\mathbf{Y}}$ position of the `SMS::Payload` changes much more slowly. Figure 32, shows the tension in the cable oscillating and transitioning into slack states, like a mass on an elastic.

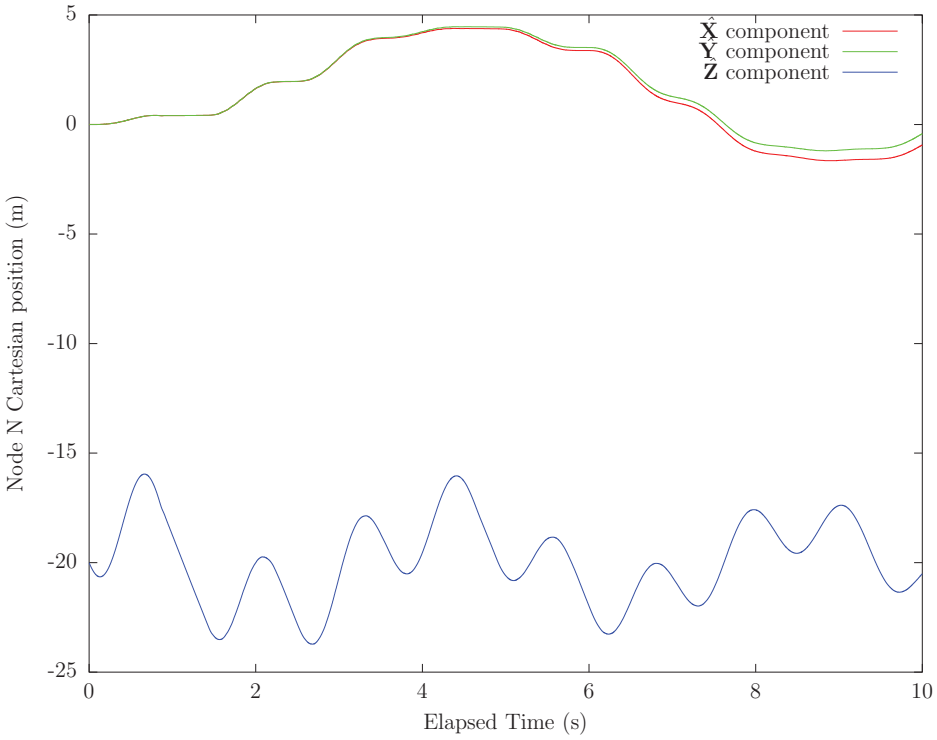


Figure 31: *The position of the cable's node N over time.*

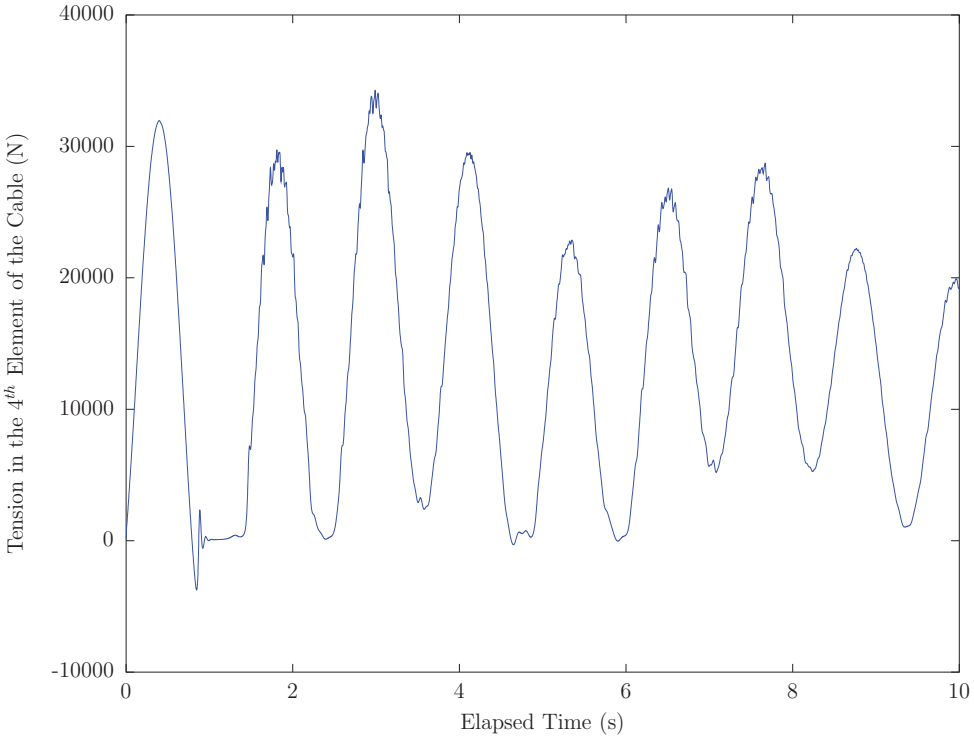


Figure 32: *The tension in the 4th element of the cable over time.*

4.1.5 Validation Data from Literature

A qualitative evaluation of an earlier version of the cable model that is presented here is given in [5]. In this publication an experiment is performed in which the top node of a 4.12m length ROV tether is oscillated in a water filled basin. Video footage of the maneuver was captured and compared with an animation of simulation results of the tether as shown in Figure 33 and Figure 34.

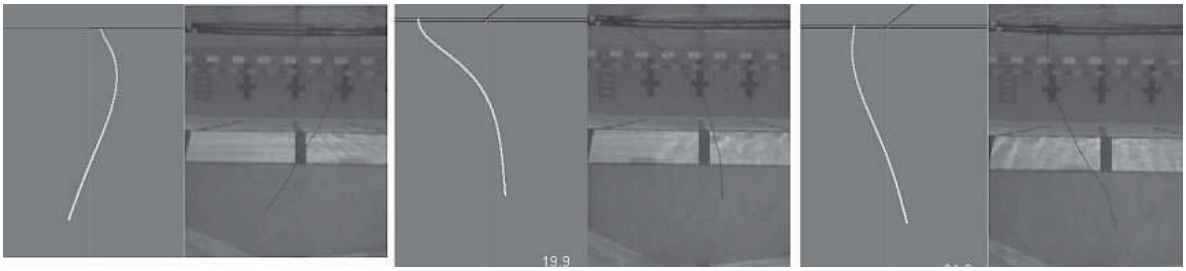


Figure 33: *A three frame comparison of the cable model simulation results (animations on left) and pool test results (pictures on right)*

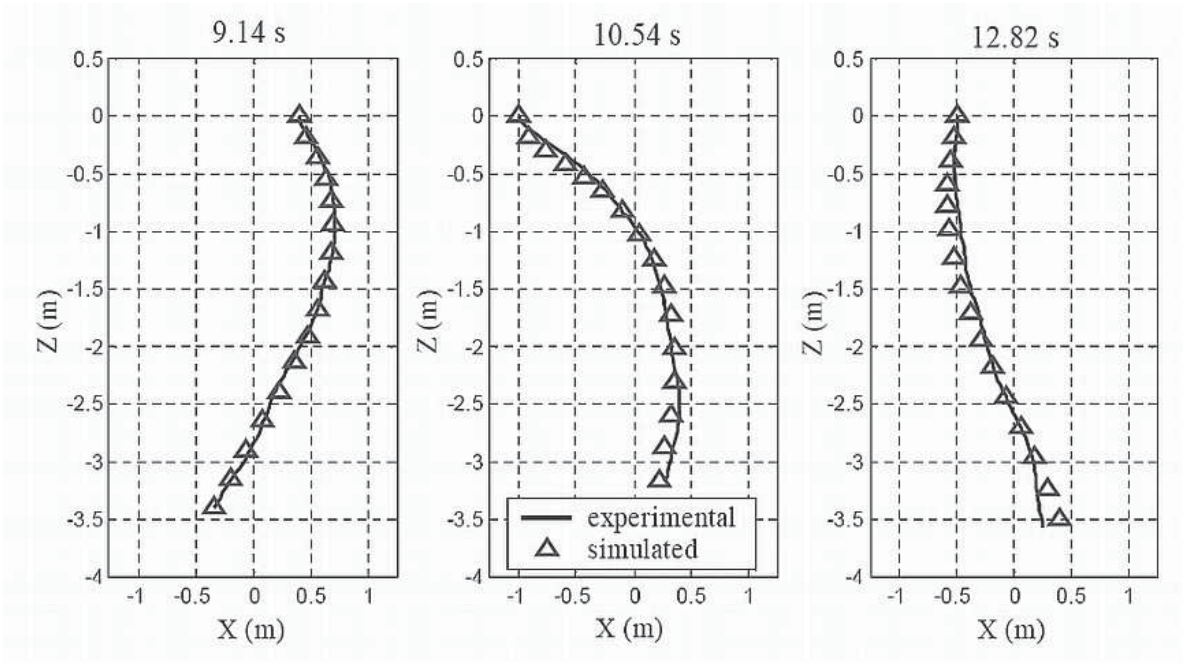


Figure 34: *This figure shows data gathered by scaling the video frames shown in Figure 33, this was done in an effort to quantify the results from those tests. The small discrepancies between the simulated and tested results is due to permanent plastic deformations in the umbilical that was tested.*

4.2 Winch Class Validation

All of the SMS::Cables simulated here have the following physical properties set in the *.ini* file:

```
$EA 8.0e5
$EI 0
$GJ 0
$rho_cable 7700.0
$dia_cable 0.005

$0|IC 0 0 0 0 0 0
$1|IC -2.434652e-001 2.915689e-001 0 0 8.295825e-003 -1.000028e+001
$2|IC -4.996554e-001 5.767018e-001 0 0 1.569803e-002 -2.000128e+001
$3|IC -4.996554e-001 5.767018e-001 0 0 1.569803e-002 -3.000128e+001

$ElementLengths 10 10 10
```

4.2.1 Constant Tension Winch Control Test with Oscillating Load Perturbation at Free Node

4.2.1.1 Motivation

The purpose of this test is to show that even with an oscillating free node load, the tension in the SMS::Cable is held at the tension setpoint by the SMS::Winch tension controller.

4.2.1.2 Experimental Setup

To verify that the winch tension controller was functioning properly, a 30 meter long axially stiff SMS::Cable with three elements was created with a SMS::Winch attached to its node 0 (see Figure 35). A 200 kg SMS::Payload was attached to the SMS::Cable's Node N position. An oscillating external load was then applied to the SMS::Payload of:

$$\mathbf{f}(t_{elapsed}) = \begin{bmatrix} 0 \\ 0 \\ 250\sin(\frac{2\pi}{5}t_{elapsed}) \end{bmatrix} \tag{108}$$

The tension in the SMS::Cable over the simulation period of 10 seconds, at the element between node 0 and node 1, was verified against its tension control set point of 2006.1 N, which is equal to the weight of the payload and the cable.

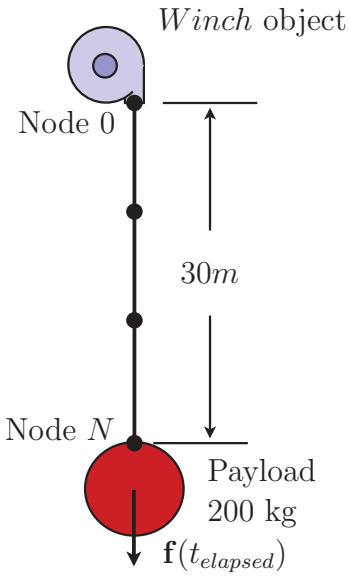


Figure 35: *The experimental setup for the winch constant tension and constant velocity tests.*

4.2.1.3 Simulation Results

In response to the oscillating force applied to the `SMS::Payload`, the `SMS::Winch` pays out and pays in cable, as shown in Figure 36. Figure 37 shows the tension in the first element of the `SMS::Cable` over the course of the 10 second simulation. There were some start up transient effects due to the action of the PID controller, but the commanded tension of 2006.1 N was maintained throughout the remaining simulation.

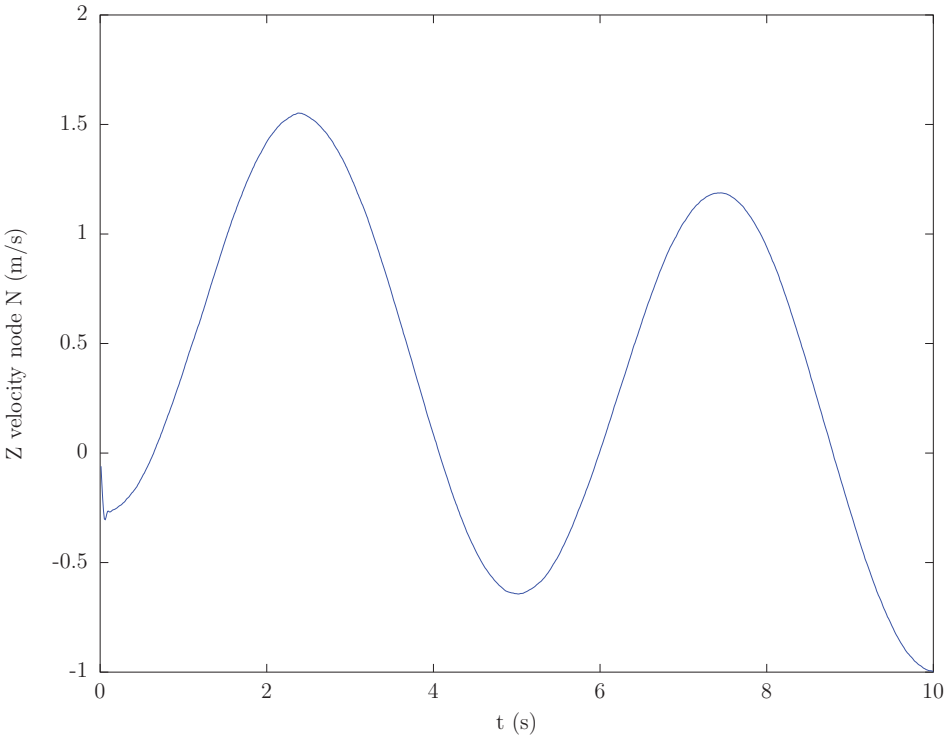


Figure 36: *Shows the winch payout velocity during the constant tension winch test.*

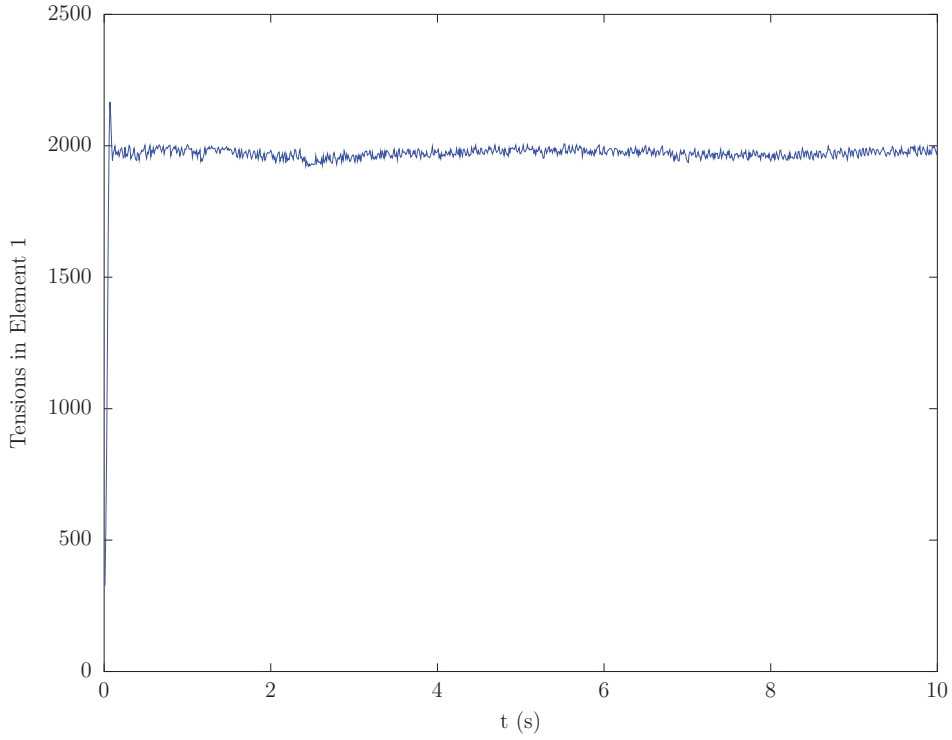


Figure 37: *The figure shows the tension in the cable during the constant tension test. The winch in this test is responding to tension changes in the cable using a PID controller; the winch pays out and pays in to increase or decrease the tension, as shown in Figure 36.*

4.2.2 Varying Tension Setpoint Winch Test

4.2.2.1 Motivation

The purpose of this test is to demonstrate that the `SMS::Winch` will maintain the tension at set point of a tension controller when the tension setpoint is varied sinusoidally through time.

4.2.2.2 Experimental Setup

Using the test setup from the constant tension test (see Section 4.2.1), without the oscillating load on the `SMS::Payload`, a simulation was performed on the `SMS::Winch` controller where tension setpoint was varied over time as:

$$f_{setpoint}(t_{elapsed}) = 2006.1 + 250\sin(\frac{2\pi}{5}t_{elapsed}) \tag{109}$$

The simulated tension in the `SMS::Cable` is then compared against the tension setpoint.

4.2.2.3 Simulation Results

Figure 38 shows the tension in the first element of the `SMS::Cable` over the course of the 20 second simulation. Figure 39 shows the `SMS::Winch` payout velocity of the `SMS::Cable` over the course of the 20 second simulation. There were some start up transient effects due to the initial action of the PID controller, but the commanded tension was maintained throughout the remaining simulation.

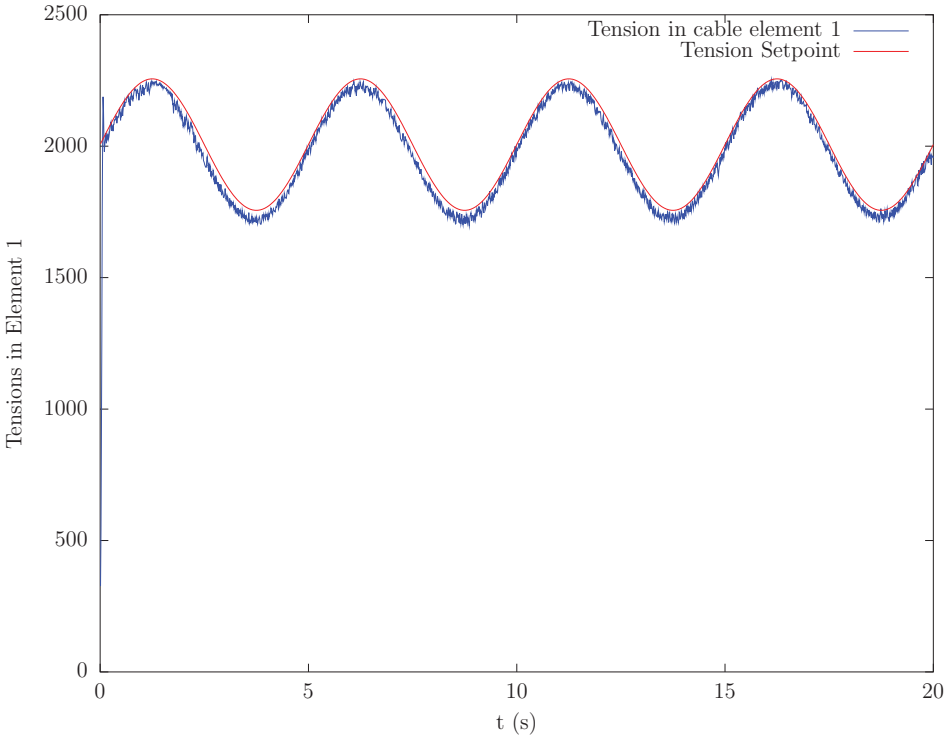


Figure 38: *The tension in the first element of the cable with a time varying tension setpoint of a winch tension controller.*

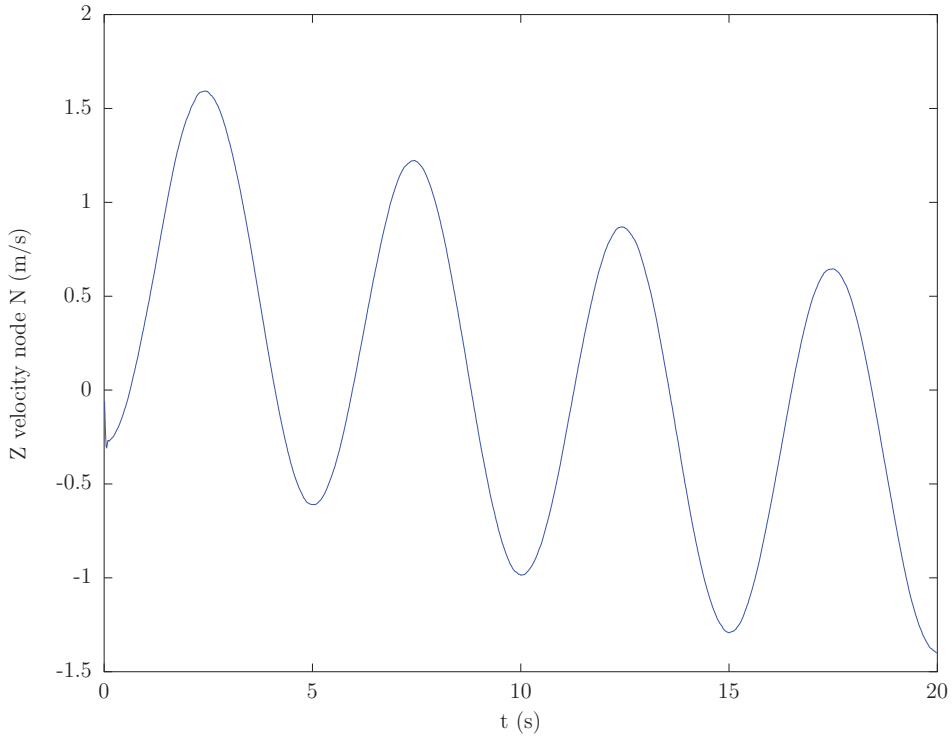


Figure 39: *The winch payout velocity for the varying tension setpoint test is shown. Note the payout velocity required to follow the tension setpoint changes as the total length and mass of the cable is increased.*

4.2.3 Varying Velocity Setpoint Winch Test

4.2.3.1 Motivation

The purpose of this test is to demonstrate that the `SMS::Winch` will maintain the payout rate dictated by the set point of the `SMS::Winch` velocity controller when the velocity setpoint is varied sinusoidally through time.

4.2.3.2 Experimental Setup

To verify that the `SMS::Winch` velocity controller was functioning properly, a 30 meter long axially stiff `SMS::Cable` with three elements was created with a `SMS::Winch` attached to its node 0 (see Figure 35). The `SMS::Winch` object velocity set point was then set to:

$$\mathbf{v}^{(Winch)}(t_{elapsed}) = \begin{bmatrix} 0 \\ 0 \\ \sin(\frac{2\pi}{5} t_{elapsed}) \end{bmatrix} \tag{110}$$

at every output cycle (0.01 seconds of elapsed time).

The velocity of the `SMS::Cable`'s last node over the simulation period of 20 seconds was compared against its velocity control set points from equation 110. Because the velocity control set points are sinusoidal, the velocity period of oscillation and amplitude of the `SMS::Cable`'s node N were compared against that of the `SMS::Winch`'s setpoint.

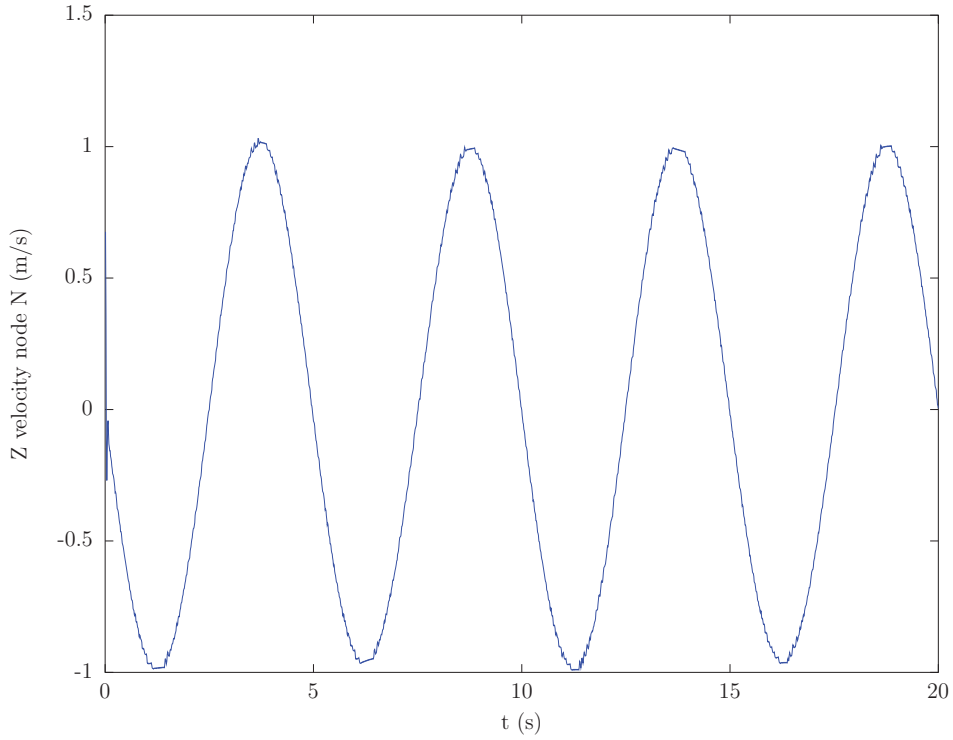


Figure 40: *The velocity of a the cable's last node whose winch was instructed to payout the cable at an oscillating velocity.*

4.2.3.3 Simulation Results

Figure 40 shows the velocity of the Nth node of the `SMS::Cable` due to the commanded `SMS::Winch` pay out velocity. Since the cable was hanging vertically, the velocity of the Nth node of the `SMS::Cable` matches the `SMS::Winch` velocity setpoint. The expected and simulated period of oscillation and amplitude of oscillation can be found in Table 6.

Output Variable	Expected	Simulated
$\tau_{vec}^{(1)}$	5.0	4.9648
A	1.0	1.0985

Table 6: *Expected and Simulated Cable Velocity Period and Amplitude of Oscillation at Node 1*

4.2.4 Constant Velocity Winch Control Test with Oscillating Load Perturbation at Free Node

4.2.4.1 Motivation

The purpose of this test is to show that even with an oscillating free node load, the payout velocity of the `SMS::Cable` is held at the velocity setpoint of the `SMS::Winch`'s velocity controller.

4.2.4.2 Experimental Setup

To show that the `SMS::Winch`'s velocity controller is able to maintain a user specified payout velocity setpoint, a 30 meter long axially stiff `SMS::Cable` with three elements was created with a `SMS::Winch` attached to node 0 (see Figure 35). In addition, an oscillating load was applied to node N of the `SMS::Cable` to see if the `SMS::Winch` controller could manage the tension disturbances. The `SMS::Winch` object velocity set point is then set to a constant $\mathbf{v}^{(Winch)}(t_{elapsed}) = 1$. An oscillatory load was applied to the `SMS::Cable` by attaching a `SMS::Payload` with a mass of 1 kg at the cable's node N and then applying a force to the `SMS::Payload` of $\mathbf{f} = [0, 0, -40 + 30\sin(\frac{2\pi}{5}t), 0, 0, 0]^T$.

4.2.4.3 Simulation Results

Figure 41 shows both the payout velocity of the `SMS::Winch` and of the Nth node of the `SMS::Cable`. A positive pay out rate represents the adding length to the `SMS::Cable`, which was negated and overlayed here over the node velocity to show the difference between the Node N velocity and the winch payout velocity. The velocity of the Nth node of the `SMS::Cable` matches closely the `SMS::Winch` payout velocity. Figure 42 shows the tension in the first element of the `SMS::Cable` due to the payout speed and the disturbance load applied to the `SMS::Payload`. In Figure 42, the mean tension increases due to the increasing length, and therefore increased hanging weight, of the `SMS::Cable`. Take note that the acceleration and deceleration limits were set to +/- 0.1 m/s^2.

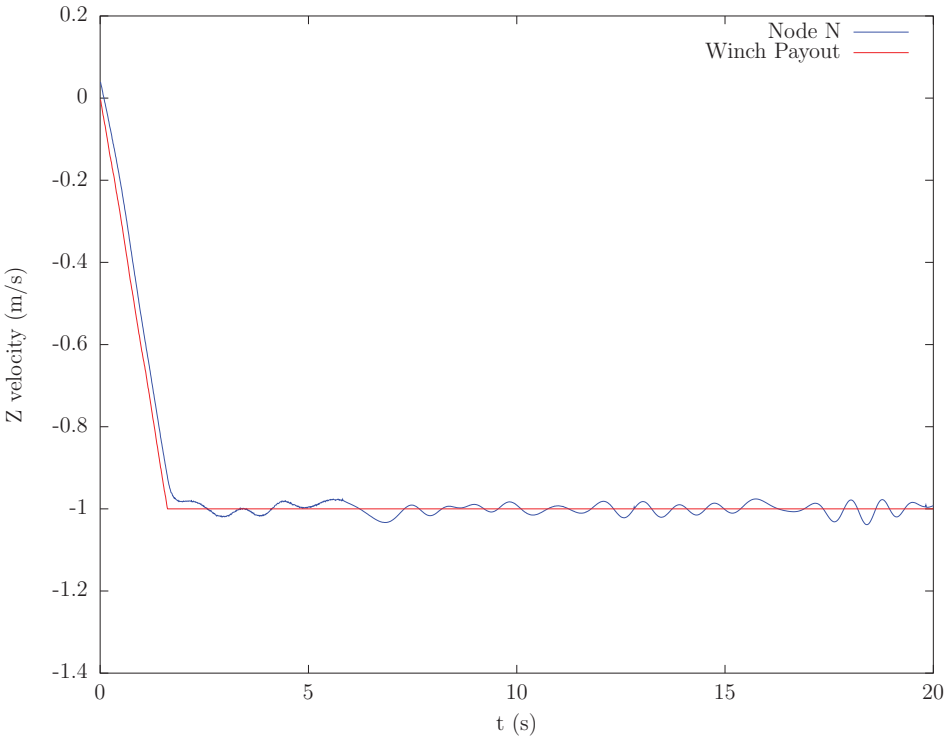


Figure 41: *The cable payout velocity for the winch with an applied oscillatory load on the cable.*

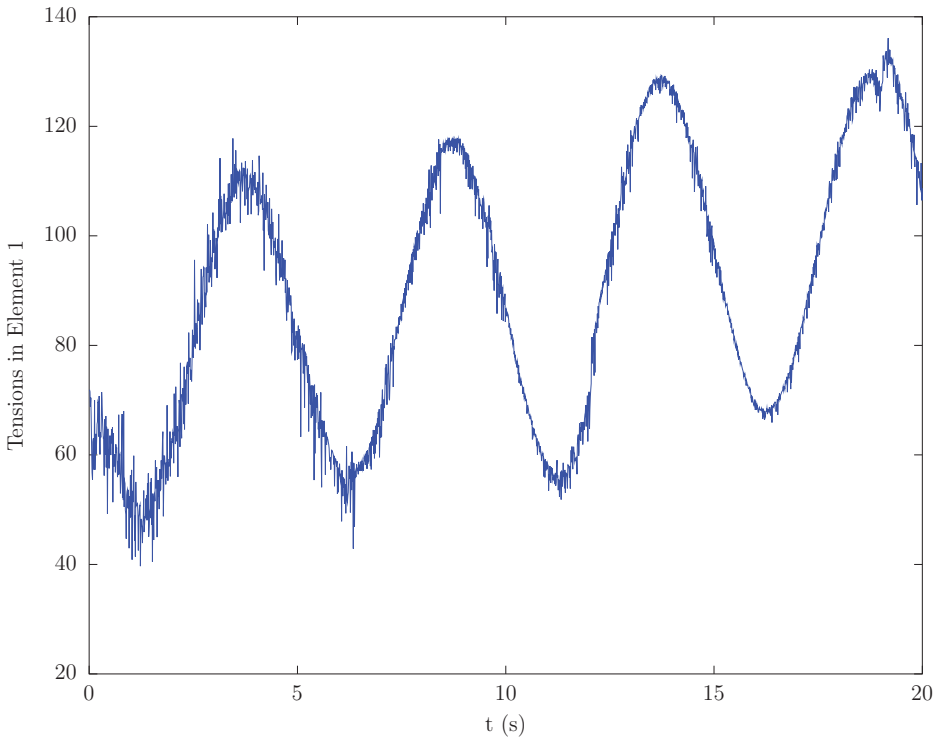


Figure 42: *The tension in the cable where the winch tries to maintain payout velocity while an oscillatory load is applied to the cable.*

4.3 Payload Class Validation

4.3.1 Linear Acceleration

4.3.1.1 Motivation

The purpose of this test is to compare the simulated `SMS::Payload`'s displacement after an elapsed time under an applied constant linear force against the expected analytical displacement.

4.3.1.2 Experimental Setup

A force of ${}^{E}\mathbf{F} = [0.01, 0.01, 0.01, 0.0, 0.0, 0.0]^{T}$ is applied to the `SMS::Payload` causing an acceleration (see Figure 43). The box has a mass of 5 kg and side lengths of 2.45 m. After 20 seconds of time has elapsed, the displacement of the payload is measured and compared against expected analytical results:

$$ {}^{E}\mathbf{p}_{cg}(t) = {}^{E}\mathbf{p}_{cg}(0) + {}^{E}\mathbf{v}_{cg}(0)t + 0.5 {}^{E}\mathbf{M}^{-1} {}^{E}\mathbf{F}t^{2} \tag{111} $$

where ${}^{E}\mathbf{v}_{cg} = {}^{E}\mathbf{R}_{B} {}^{B}\mathbf{v}_{cg}$ is the spatial vector of the velocity of the rigidbody expressed with respect to the frame E and ${}^{E}\mathbf{R}_{B}$ is a 6×6 matrix that rotates a spatial vector from one frame to another and is expressed as:

$$ {}^{E}\mathbf{R}_{B} = \left| \begin{array}{cc} \mathbf{R}_{1} & 0_{3\times3} \\ 0_{3\times3} & \mathbf{R}_{2} \end{array} \right| \tag{112} $$

where $\mathbf{R}_{2} = \mathbf{R}_{1}$.

4.3.1.3 Simulation Results

Table 7 shows the expected final spatial velocity vector and the simulated final spatial velocity vector.

Variable	Expected Velocity	Simulated Velocity
${}^{E}\upsilon_{cg}$	$\begin{bmatrix} 0.4 \\ 0.4 \\ 0.4 \\ 0.0 \\ 0.0 \\ 0.0 \end{bmatrix}$	$\begin{bmatrix} 0.40000000000000790 \\ 0.40000000000000790 \\ 0.40000000000000790 \\ 0.00000000000000000 \\ 0.00000000000000000 \\ 0.00000000000000000 \end{bmatrix}$

Table 7: *The expected and simulated velocity for a* `SMS::Payload` *with a force applied about each linear DOFs.*

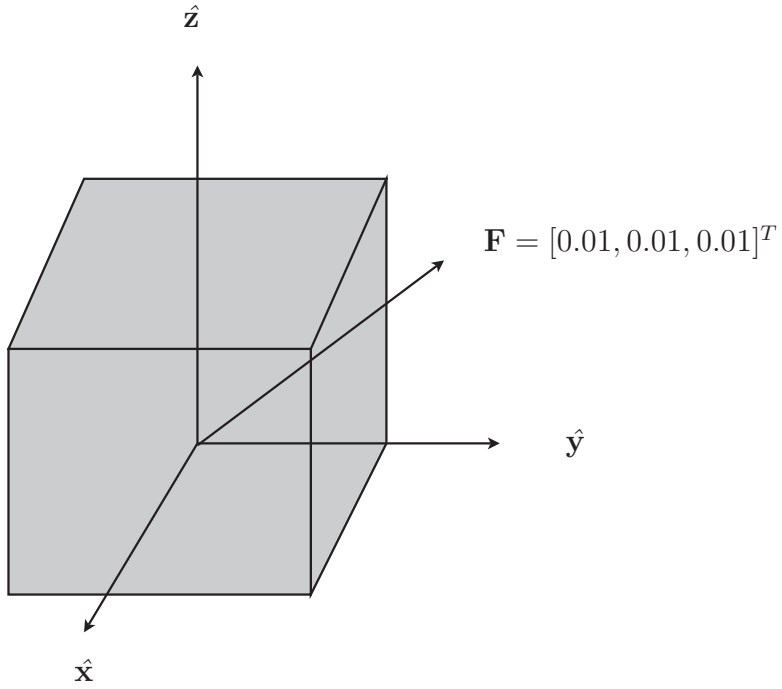


Figure 43: *The forcing of the rigid body for the linear acceleration test.*

4.3.2 Angular Acceleration

4.3.2.1 Motivation

To compare the simulated `SMS::Payload`'s angular velocity after elapsed time with an applied constant moment against an analytical solution.

4.3.2.2 Experimental Setup

A moment of ${}^{E}\mathbf{F} = [0.0, 0.0, 0.0, 0.01, 0.01, 0.01]^T$ is applied to the `SMS::Payload` causing an acceleration (see Figure 44). After 20 seconds of time has elapsed, the angular velocity of the payload is measured and compared against expected analytical results:

It should be noted that orientation here is described with Euler angles about the axes of a non-orthogonal frame. In order to validate the angular acceleration, the angular velocity vector about the body-fixed frame is transformed to the inertial frame (both orthogonal frames), which facilitates validation.

$$ {}^{E}\mathbf{v}_{cg}(t) = {}^{E}\,\mathbf{v}_{cg}(0) + {}^{E}\,\mathbf{M}^{-1}\ {}^{E}\mathbf{F}t \tag{113} $$

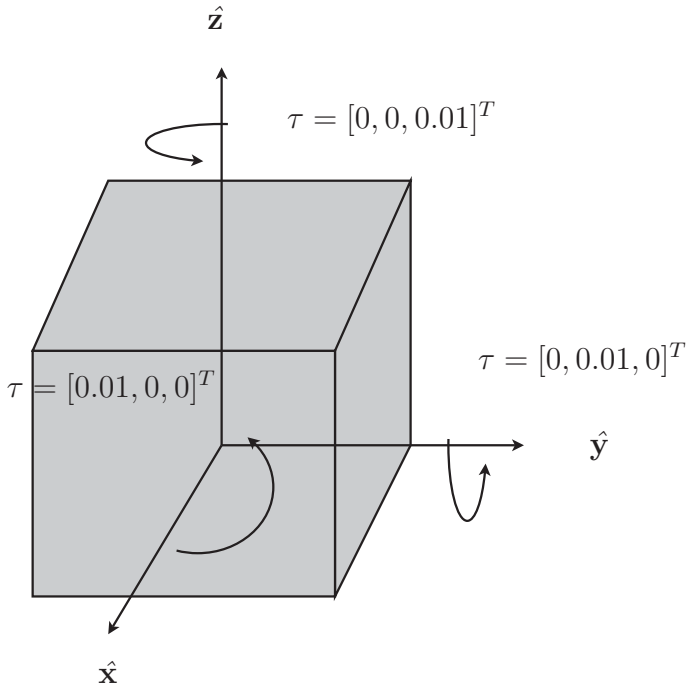


Figure 44: *The forcing of the rigid body for the angular acceleration test.*

4.3.2.3 Simulation Results

Table 8 shows the expected final spatial velocity vector and the simulated final spatial velocity vector.

Variable	Expected Velocity	Simulated Velocity
${}^{E}\mathcal{V}_{cg}$	$\begin{bmatrix} 0.0 \\ 0.0 \\ 0.0 \\ 0.04 \\ 0.04 \\ 0.04 \end{bmatrix}$	$\begin{bmatrix} 0.00000000000000000 \\ 0.00000000000000000 \\ 0.00000000000000000 \\ 0.039997500102352207 \\ 0.039997500018007048 \\ 0.039997500041729239 \end{bmatrix}$

Table 8: *The expected and simulated velocity for a* `SMS::Payload` *with a force applied about each angular DOF.*

4.3.3 Combined Linear and Angular Acceleration

4.3.3.1 Motivation

The purpose of this test is to compare the `SMS::Payload`'s simulated linear displacement and angular velocity after an elapsed time, with an applied constant linear force and moment, against an analytical solution.

4.3.3.2 Experimental Setup

A force and moment of ${}^{E}\mathbf{F} = [0.01, 0.01, 0.01, 0.01, 0.01, 0.01]^{T}$ is applied to the `SMS::Payload` causing an acceleration (see Figure 45). After 20 seconds of time has elapsed, the linear and angular velocity of the `SMS::Payload` is measured and compared against expected analytical results.

$$ {}^{B}\upsilon_{cg}(t) =^{E} \upsilon_{cg}(0) + =^{E} \mathbf{M}^{-1}\ {}^{E}\mathbf{F}t \tag{114} $$

It should be noted that orientation here is described as angles about the axes of a non-orthogonal frame; therefore, in order to validate the angular acceleration, the angular velocity vector described about the body-fixed frame is transformed to the inertial frame which is then used to validate the rigid body model.

4.3.3.3 Simulation Results

Table 9 shows the expected final spatial velocity vector and the simulated final spatial velocity vector.

Variable	Expected Velocity	Simulated Velocity
${}^{E}\upsilon_{cg}$	$\begin{bmatrix} 0.04 \\ 0.04 \\ 0.04 \\ 0.04 \\ 0.04 \\ 0.04 \end{bmatrix}$	$\begin{bmatrix} 0.039999749999997655 \\ 0.039999749999997627 \\ 0.039999749999997662 \\ 0.039999750000003116 \\ 0.039999750000003428 \\ 0.039999750000003199 \end{bmatrix}$

Table 9: *The expected and simulated velocity for a `SMS::Payload` with a force applied about each of its 6 DOF.*

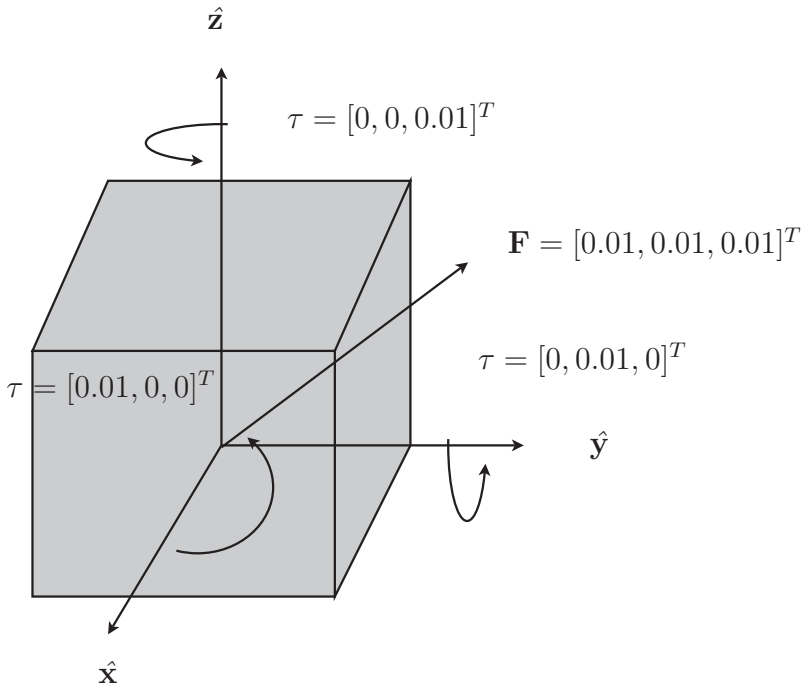


Figure 45: *The forcing of the rigid body for the linear and angular acceleration test.*

4.3.4 Payload−Cable Attach

4.3.4.1 Motivation

The purpose of this test is to show that an `SMS::Payload` can be attached to an `SMS::Cable` and have the `SMS::Cable` support the weight of the `SMS::Payload`.

4.3.4.2 Experimental Setup

The `SMS::Payload` is attached to the free end of an `SMS::Cable` whose other end is fixed in space at $[0, 0, 0]^T$. The simulation is run for 60 seconds and the final resting position of the payload as well as the tension in the cable are compared with expected results.

The tension in the `SMS::Cable` at the final element should be:

$$F_t = mg \tag{115}$$

where $g = 9.81$ and $m = 5$ kg and the final position in $\hat{\mathbf{Z}}$ should be equal to the stretched length of the cable. Since the mass of the `SMS::Payload` is relatively small, the strain in the cable will be relatively small, thus; the unstreched `SMS::Cable` length should be approximately the same as the stretched cable length.

4.3.4.3 Simulation Results

Table 10 compares the simulated and expected steady state tensions in the `SMS::Cable` and the final resting position of the `SMS::Payload`.

Variable	Expected Results	Simulated Results
F_t	49.5	49.06894
$P_{\hat{\mathbf{z}}}$	-20.0	-20.00108

Table 10: *Expected and simulated `SMS::Cable` tension at the attachment point and the final resting position of the `SMS::Payload`.*

4.3.5 Payload−Cable Pendulum

4.3.5.1 Motivation

The purpose of this test is to attach the `SMS::Payload` to the bottom end of a vertical `SMS::Cable` and measure period of oscillation induced from a small initial lateral displacement and compare it against analytical.

4.3.5.2 Experimental Setup

A `SMS::Payload` is attached to the free end of a `SMS::Cable` whose other end is fixed in space at $[0, 0, 0]^T$. The `SMS::Payload` moved such that it has a small lateral displacement from vertical (see Figure 46). The simulation is executed for 60 seconds. The position of the `SMS::Payload` through time is processed through *pwelch*, a signal processing tool for Matlab and GNU/Octave to obtain the oscillation period.

For small deflections, the analytical period of oscillation for a lumped mass at the end of a massless cable is:

$$t_{Pendulum} = 2\pi\sqrt{\frac{L}{g}} \tag{116}$$

where $t_{Pendulum}$ is the period of oscillation, L is the length of the cable and g is the gravitational acceleration.

4.3.5.3 Simulation Results

Table 11 compares the simulated and expected periods of oscillation of `SMS::Cable` and a `SMS::Payload`.

Variable	Expected Results	Simulated Results
$t_{Pendulum}$	8.97	8.85

Table 11: *Expected and simulated period of oscillation of pendulum with* `SMS::Cable` *and* `SMS::Payload`*.*

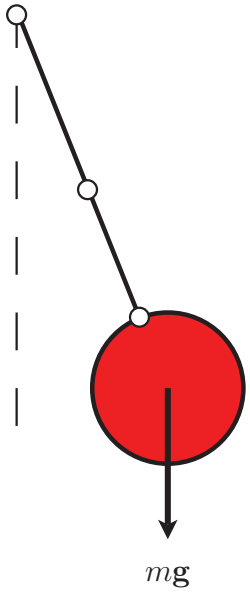


Figure 46: *Diagram of an experiment to test the period of oscillation, a function of gravity and cable length, of a pendulum.*

4.3.6 Payload−Cable Torsion

4.3.6.1 Motivation

The purpose of this test is to show that an `SMS::Payload` attached to the bottom end of a vertical `SMS::Cable`, with an initial `SMS::Payload` angular velocity about the $\hat{\mathbf{Z}}$ axis, the system will have a torsional period of oscillation equal to the analytical solution.

4.3.6.2 Experimental Setup

An `SMS::Payload` is attached to the free end of a cable whose other end is fixed in space at $[0, 0, 0]^T$. The payload is given an initial angular velocity about the vertical axis, $\hat{\mathbf{Z}}$ (see Figure 47). The simulation is executed for 60 seconds. The yaw orientation of the payload through time is processed with *pwelch*, a signal processing tool for Matlab and GNU/Octave to obtain the oscillation period.

The period of oscillation follows from the solution to a torsional mass-spring-damper equation:

$$t_{Torsional} = 2\pi\sqrt{\frac{LI_z}{GJ}} \tag{117}$$

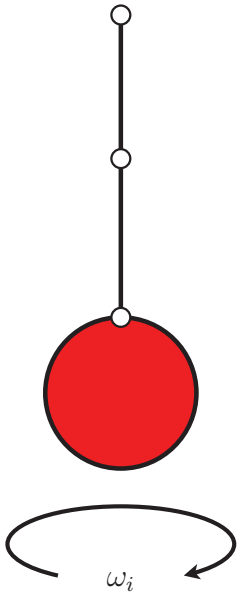


Figure 47: *Diagram of an experiment to test the period of oscillation of a mass twisting about the axis of a taut cable which it is attached to.*

where $t_{Torsional}$ is the period of oscillation, L is the length of the cable, GJ is the torsional stiffness of the cable, and I_z is the total mass moment of inertia about the axis of rotation, $\hat{\mathbf{Z}}$.

4.3.6.3 Simulation Results

Table 12 compares the simulated and expected periods of torsional oscillation a pendulum consisting of a `SMS::Cable` and a `SMS::Payload`.

Variable	Expected Results	Simulated Results
$t_{Torsion}$	6.2832	6.2864

Table 12: *Expected and simulated `SMS::Cable`-`SMS::Payload` period of oscillation in torsion.*

4.4 BoomCrane Class Validation

4.4.1 Revolute Joint Test

4.4.1.1 Motivation

The purpose of this validation test is to ensure that links with revolute proximal joints function properly. A single link with an off-CG revolute proximal joint oscillates under gravity. The period of oscillation is compared against the analytical equation for period of oscillation of a pendulum.

4.4.1.2 Experimental Setup

An articulated body (see Figure 48) is defined by the D-H Parameters given in Table 13.

Link id	α_i	a_i	d_i	θ_i
1	0	2	0	-89.9*

Table 13: *D-H Parameters for the revolute joint validation test.*

The origin of the body-fixed reference frame is coincident with the origin of the Earth-fixed reference frame at p, and the orientation of the body-fixed frame is rotated by 90 degrees about the common $\hat{\mathbf{x}}$-axis from the Earth-fixed frame. This state can be represented by the array $\check{\mathbf{P}}^{(1)}_{E\to p} = \left[\begin{array}{cccccc} 0 & 0 & 0 & \frac{\pi}{2} & 0 & 0 \end{array}\right]^T$.

The `SMS::Boomcrane`'s only link's rigid body, a rectangular cuboid with a mass of 10 kg, density of 7778 kg/m^3 and dimensions of $L_x = 2$ m, $L_y = 0.025$ m, and $L_z = 0.025$ m has the following mass properties:

$$\mathbf{M}_{CG} = \left[\begin{array}{cccccc} 10 & 0 & 0 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 & 0 & 0 \\ 0 & 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.001 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3.334 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3.334 \end{array}\right] \tag{118}$$

and the position of the rigid body's center of gravity relative to p, $\check{\mathbf{P}}^{(i)}_{p\to CG}$, can be represented by the vector $\check{\mathbf{P}}^{(i)}_{p\to CG} = \left[\begin{array}{c} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{array}\right]$.

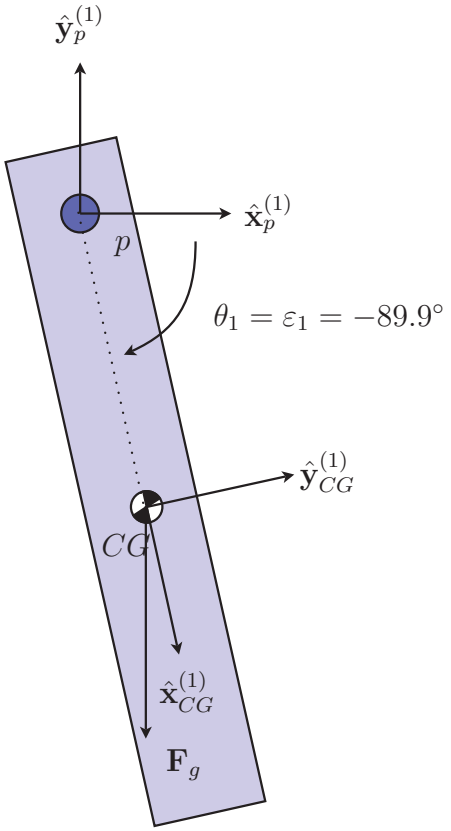


Figure 48: *The experimental setup for the revolute joint validation test. Shown here with no joint displacement.*

Gravity will impose a force at the CG of the rigid body which will cause the link to pendulate about its revolute joint axis. The expected period of oscillation of a rigid body about an off-CG revolute joint is:

$$\tau_{rev} = 2\pi\sqrt{\frac{\mathbf{I}_p^{(1)}}{m^{(1)}gL}} \tag{119}$$

where $\mathbf{I}_p^{(i)}$ is the moment of inertia of the rigid body about its proximal joint p frame, m is the mass of the rigid body, g is the gravity acceleration, and L is the distance between the axis of rotation and the CG.

4.4.1.3 Simulation Results

The simulated period of oscillation of a single-link, revolute joint `SMS::BoomCrane` oscillating under gravity is presented in Table 14 along with the expected analytical results for comparision.

Variable	Expected Period	Simulated Period
τ_{rev}	$2\pi\sqrt{\frac{14.1667kgm^2}{(10kg)(9.81m/s^2)(1m)}} = 2.3153$ s	2.3076 s

Table 14: *The simulated and expected period of oscillation of a single revolute joint link under gravity.*

4.4.2 Prismatic Joint Test

4.4.2.1 Motivation

The purpose of this validation test is to ensure that links with prismatic proximal joints function properly. A single link with a prismatic proximal joint falls freely under gravity. The final position of the joint after an elapsed period of time is compared against analytical expected result.

4.4.2.2 Experimental Setup

An articulated body (see Figure 49) is defined by D-H Parameters given in Table 15.

Link id	α_i	a_i	d_i	θ_i
1	0	0	0*	0

Table 15: *D-H Parameters for the prismatic joint validation test.*

The base frame relative to the Earth-fixed frame, $\check{\mathbf{P}}^{(1)}_{E\to p}$ is made of the 3 Cartesian vector position components and 3 Euler angles components that define its orientation. In this case $\check{\mathbf{P}}^{(1)}_{E\to p} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T$. This defines the first joint's frame.

The `SMS::Boomcrane`'s only link's rigid body, a rectangular cuboid with a mass of 10 kg, density of 7778 kg/m^3 and dimensions of $L_x = 2$ m, $L_y = 0.025$ m, and $L_z = 0.025$ m has the following mass properties:

$$\mathbf{M}_{CG} = \begin{bmatrix} 10 & 0 & 0 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 & 0 & 0 \\ 0 & 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.001042 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3.334 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3.334 \end{bmatrix} \tag{120}$$

The position of the rigid body, $\check{\mathbf{P}}^{(1)}_{p\to CG}$, made of the 3 Cartesian position components and 3 Euler angles that define the orientation is defined as:

$$\check{\mathbf{P}}^{(1)}_{p\to CG} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{121}$$

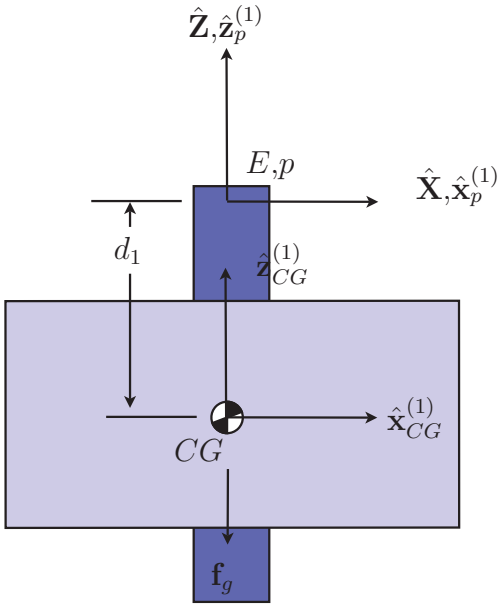


Figure 49: *The experimental setup for the prismatic joint validation test. Shown here with a joint displacement of* d_1.

Gravity will impose a force at the CG of the rigid body which will cause the link to fall along its prismatic joint axis. The joint position after an amount of time has elapsed is:

$$\varepsilon_1(t_{elapsed}) = \dot{\varepsilon}_1(0)\ t_{elapsed}\ +\ 0.5\ g\ t_{elapsed}^2 \tag{122}$$

where g is the gravitational acceleration, $t_{elapsed}$ is the elapsed time and ε_1 is the joint displacement.

4.4.2.3 Simulation Results

The simulated displacement of a single-link, prismatic joint `SMS::BoomCrane` falling under gravity is presented in Table 16 along with the expected analytical results for comparision.

Variable	Expected Joint Displacement	Simulated Displacement
$\varepsilon_p^{(1)}(30)$	$(0.5)\ (9.81\ \text{m/s}^2)\ (30\text{s})^2 = 4414.5\ \text{m}$	4414.5 m

Table 16: *The simulated and expected joint displacement for the Prismatic Joint Test.*

4.4.3 Double Pendulum Test

4.4.3.1 Motivation

The motivation of this experiment is to test a more complex dynamic motion with multiple degrees of freedom. In this case, a double pendulum is simulated with the use of two dynamic revolute joints.

4.4.3.2 Experimental Setup

An articulated body (see Figure 50) is defined by the D-H Parameters given in Table 17.

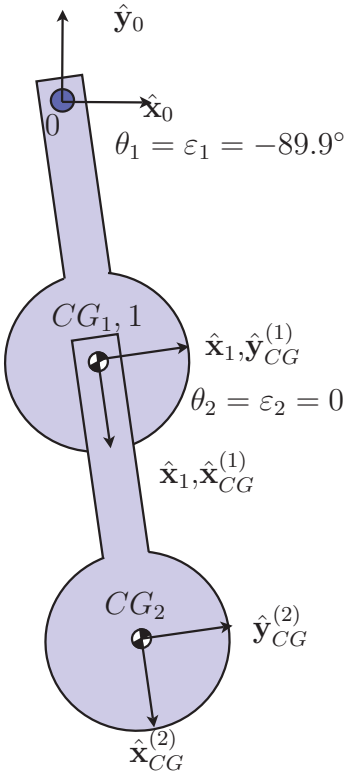


Figure 50: *The experimental setup for the pendulum validation test.*

Link id	α_i	a_i	d_i	θ_i
1	0	10	0	-89.9*
2	0	7	0	0*

Table 17: *D-H Parameters for the pendulum boom crane joint validation test.*

The base frame relative to the Earth-fixed frame, $\check{\mathbf{P}}^{(1)}_{E\to p}$, made of the 3 Cartesian position components and 3 Euler angles that define the orientation is $\begin{bmatrix} 0 & 0 & 0 & \frac{\pi}{2} & 0 & 0 \end{bmatrix}^T$.

This defines the first joint's frame.

Each rigid link is modeled as a sphere whose center is at the CG. Each link has a mass of 10 kg and cylindrical rod connecting it to the center of rotation of the first joint. The sphere's radius is 0.06745 m and thus has the density of steel. Since the rod mass is much less than the sphere, it is assumed to have no mass contribution to the link. The rigid-body mass properties for link 1 are:

$$\mathbf{M}_{CG}^{(1)} = \begin{bmatrix} 10 & 0 & 0 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 & 0 & 0 \\ 0 & 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.018171 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.018171 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.018171 \end{bmatrix} \tag{123}$$

and for link 2:

$$\mathbf{M}_{CG}^{(2)} = \begin{bmatrix} 10 & 0 & 0 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 & 0 & 0 \\ 0 & 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.018171 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.018171 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.018171 \end{bmatrix} \tag{124}$$

The position of the rigid body of the first link about its proximal joint frame, $\check{\mathbf{P}}_{p\to CG}^{(1)}$, is made of the 3 Cartesian vector position components and 3 Euler angles that define its orientation is defined as:

$$\check{\mathbf{P}}_{p\to CG}^{(1)} = \begin{bmatrix} 10 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{125}$$

while the position of the rigid body of the second link about its proximal joint frame, $\check{\mathbf{P}}_{p\to CG}^{(2)}$, is defined as:

$$\check{\mathbf{P}}_{p\to CG}^{(2)} = \begin{bmatrix} 7 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{126}$$

To simplify the analytical derivation, rigid body links are treated as point masses. The Lagrangian for such a 2 DOF system of point masses is described as [42]:

$$L = T - V \tag{127}$$

where

$$T = \frac{1}{2}m_1 l_1^2 \dot{\gamma}_1^2 + \frac{1}{2}m_2 l_1^2 \dot{\gamma}_1^2 + \frac{1}{2}m_2 l_2^2 \dot{\gamma}_2^2 + m_2 l_1 l_2 \dot{\gamma}_1 \dot{\gamma}_2 \hat{\mathbf{z}}_1 \cdot \hat{\mathbf{z}}_2 \tag{128}$$

$$V = m_1 g(l_1 - l_1 \cos(\gamma_1)) + m_2 g(l_1 + l_2 - l_1 \cos(\gamma_1) - l_2 \cos(\gamma_2)) \tag{129}$$

where m_1 and m_2 are the masses of the point masses for each link, g is the gravitational acceleration and l_1 and l_2 are the distance between the joints/CGs. In this analysis, γ_1 is the angle of the joint about the vertical, because this differs from our choice of frames for this articulated body experimental setup, we set $\gamma_1 = 90° - \varepsilon_1$ and $\gamma_2 = \varepsilon_2$. This allows the use of the small angle approximation for linearizing equations 128 and 129.

The equations are linearized with the assumption that when $\gamma_1, \gamma_2 << 1$ and $\hat{\mathbf{z}}_1 \cdot \hat{\mathbf{z}}_2 \approx 1$ then $\cos(\gamma)_i \approx 1 - \frac{\gamma_i^2}{2}$ and $\sin(\gamma)_i \approx \gamma_i$. Both T and V then become:

$$T \approx \frac{1}{2}m_1 l_1^2 \dot{\gamma}_1^2 + \frac{1}{2}m_2 l_1^2 \dot{\gamma}_1^2 + \frac{1}{2}m_2 l_2^2 \dot{\gamma}_2^2 + m_2 l_1 l_2 \dot{\gamma_1} \dot{\gamma_2} \tag{130}$$

$$V \approx \frac{1}{2}(m_1 g l_1 \gamma_1^2 + m_2 g l_1 \gamma_1^2 + m_2 g l_2 \gamma_2^2) \tag{131}$$

Substituting into the Euler-Lagrange equation and with the assumption that no non-potential work is done:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\gamma}_i}\right) = \left(\frac{\partial L}{\partial \gamma_i}\right) \tag{132}$$

leads to:

$$\begin{bmatrix} l_1(m_1 + m_2) & m_2 l_2 \\ l_1 & l_2 \end{bmatrix} \begin{bmatrix} \ddot{\gamma}_1 \\ \ddot{\gamma}_2 \end{bmatrix} = - \begin{bmatrix} (m_1 + m_2)g & 0 \\ 0 & g \end{bmatrix} \begin{bmatrix} \gamma_1 \\ \gamma_2 \end{bmatrix} \tag{133}$$

or

$$[\mathbf{M}]\,[\ddot{\gamma}] \; + [\mathbf{K}]\,[\gamma] = 0 \tag{134}$$

Equation 134 is the equation of motion of the 2 DOF system. Linear differential equations accept solutions in the form of [43]:

$$[\gamma] = [\mathbf{\Gamma}] e^{iwt} \tag{135}$$

and so:

$$\left(-\left[\mathbf{M}\right]\omega^2+\left[\mathbf{K}\right]\right)\left[\mathbf{\Gamma}\right]e^{i\omega t}=0 \tag{136}$$

Because $e^{i\omega t}$ cannot equal 0, equation 134 becomes:

$$\left[-\left[\mathbf{M}\right]\omega^2+\left[\mathbf{K}\right]\right]\left[\mathbf{\Gamma}\right]=0 \tag{137}$$

This is then treated as an eigenvalue problem and the natural frequencies of the system can be extracted accordingly.

$$\left[\left[\mathbf{M}\right]^{-1}\left[\mathbf{K}\right]-\omega^2\left[\mathbf{I}\right]\right]\left[\mathbf{\Gamma}\right]=0 \tag{138}$$

$$\left[\left[\mathbf{A}\right]-\lambda\left[\mathbf{I}\right]\right]\left[\mathbf{\Gamma}\right]=0 \tag{139}$$

4.4.3.3 Simulation Results

Figure 51 indicates the frequency power spectrum plot produced from analysis of the time history of the first dynamic joint motion. The two peaks indicate two strong modes of vibration that correspond to the natural frequencies of the double pendulum. The expected and simulated periods of oscillation are provided in Table 18.

Variable	Expected Period	Simulated Period
$\tau_{pendulum}(1stNaturalPeriod)$	3.104088 s	3.114829 s
$\tau_{pendulum}(2ndNaturalPeriod)$	7.662144 s	7.550230 s

Table 18: *The two first periods of oscillating of the* `SMS::BoomCrane` *double pendulum.*

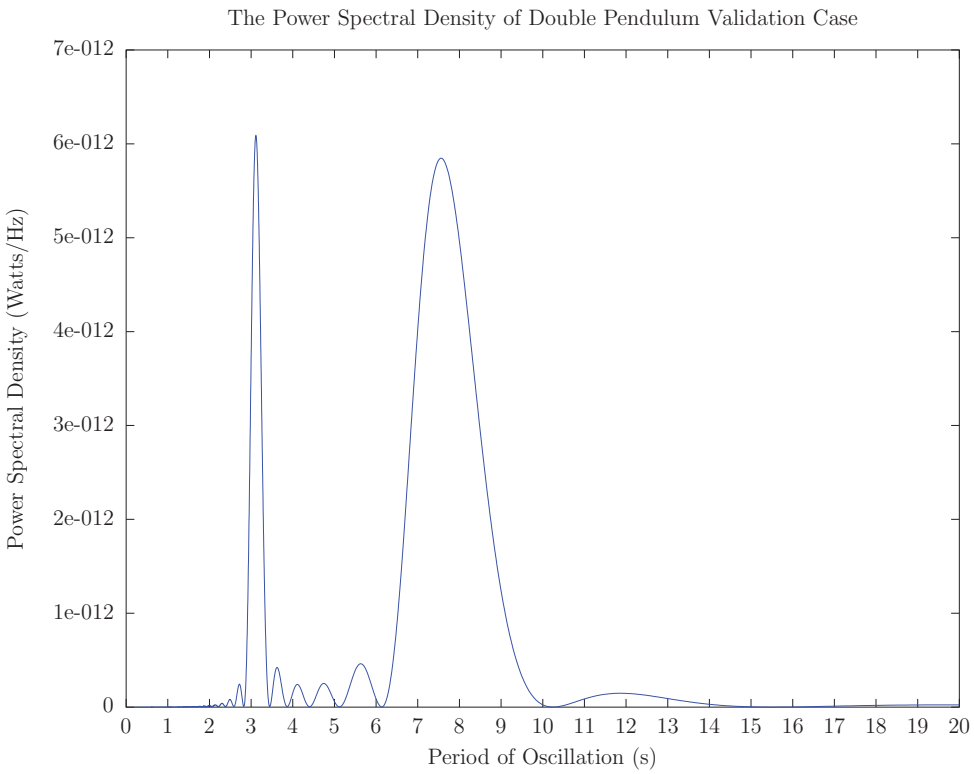


Figure 51: *The power spectral density of the oscillation of the simulated double pendulum.*

4.4.4 Centripetal Test

4.4.4.1 Motivation

The purpose of this validation test is to ensure that the kinematics of multi-link bodies generate proper centripetal accelerations. This is measured here as the force required to prevent a joint from accelerating.

4.4.4.2 Experimental Setup

An articulated body (see Figure 52) is defined D-H Parameters given in Table 19.

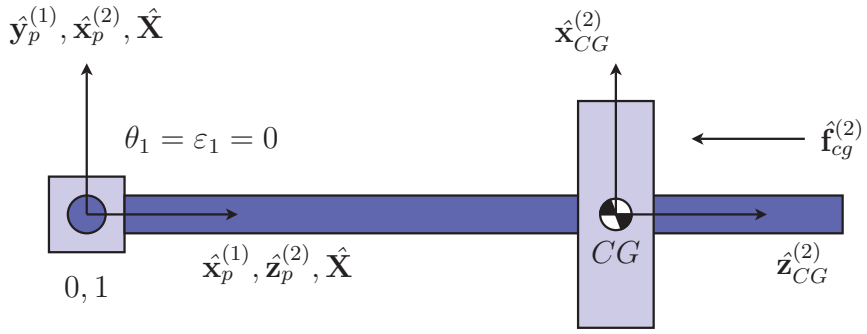


Figure 52: *The experimental setup for the centripetal, Coriolis and conservation of energy validation tests. Shown here with joint displacement of* $\theta_1 = \epsilon_1 = 0$.

Link id	α_i	a_i	d_i	θ_i
1	90	0	0	90*
2	0	0	0*	0

Table 19: *D-H Parameters for the centripetal boom crane joint validation test.*

The base frame relative to the Earth-fixed frame, $\check{\mathbf{P}}_{E\to p}^{(1)}$, is made of 3 Cartesian vector position components and 3 Euler angle components that define its orientation. It is defined here as: $\check{\mathbf{P}}_{E\to p}^{(1)} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T$, which defines the first joint's frame.

The rigid body for link 1, a rectangular cuboid with a mass of 10 kg and dimensions of

$L_x = 2$ m, $L_y = 0.025$ m, and $L_z = 0.025$ m having the following mass properties:

$$\mathbf{M}_{CG}^{(1)} = \begin{bmatrix} 10 & 0 & 0 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 & 0 & 0 \\ 0 & 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.001 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3.334 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3.334 \end{bmatrix} \tag{140}$$

and the rigid body for link 2, a disk with a mass of 10 kg, a thickness of 0.25 m, a radius of 0.0404 m, and thus has the following mass properties:

$$\mathbf{M}_{CG}^{(2)} = \begin{bmatrix} 10 & 0 & 0 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 & 0 & 0 \\ 0 & 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.082 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.056 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.056 \end{bmatrix} \tag{141}$$

The position of the rigid body of the first link about the first joint, $\check{\mathbf{P}}_{p\to CG}^{(1)}$, made of the 3 Cartesian position components and 3 Euler angles that define the orientation, is defined as:

$$\check{\mathbf{P}}_{p\to CG}^{(1)} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{142}$$

while the position of the rigid body of the second link about the second joint, $\check{\mathbf{P}}_{p\to CG}^{(2)}$, is defined as:

$$\check{\mathbf{P}}_{p\to CG}^{(2)} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{143}$$

To verify that the boom crane centripetal forces are behaving properly, the velocity of joint 1 is maintained at 1 rad/s while the force required to maintain zero displacement of joint 2 is compared against expected: $-m_{CG}^{(2)}\dot{\varepsilon}_1^2|\mathbf{P}_{1\to CG^{(2)}}|$, where $|\mathbf{P}_{1\to CG^{(2)}}|$ is the magnitude of the position of link 2's rigid body to link 1's proximal joint frame, $\dot{\varepsilon}_1$ is link 1's proximal joint actuation rate, and $m_{CG}^{(2)}$ is the mass of link 2's rigid body.

4.4.4.3 Simulation Results

Table 20 shows the expected and simulated centripetal force acting on the 2nd link of the `SMS::BoomCrane`.

Variable	Expected Centripetal Force	Simulated Simulated Centripetal Force
f	$-m_{CG}^{(2)}\dot{\varepsilon}_1^2 \lvert \hat{\mathbf{x}}_p^{(1)} \cdot \mathbf{P}_{1 \to CG^{(2)}} \rvert$ $= -(10kg)(1\text{rad/s})^2(1\text{ m}) = \text{-10 N}$	-10.000010 N

Table 20: *The expected and simulated centripetal forces for an* `SMS::BoomCrane`.

4.4.5 Coriolis Test

4.4.5.1 Motivation

The purpose of this validation test is to ensure that the kinematics of multi-link bodies generate proper Coriolis accelerations.

4.4.5.2 Experimental Setup

The articulated body here is defined as in Section 4.4.4 except joint 2 is not restrained.

To verify that the boom crane Coriolis effect is accounted for properly, the velocity of joint 1 is maintained at 1 rad/s while the acceleration of joint 2 is compared against expected: $2\dot{\varepsilon}_1\dot{\varepsilon}_2$.

4.4.5.3 Simulation Results

Table 21 shows the expected and simulated Coriolis acceleration experienced by the 2nd link of the `SMS::BoomCrane`.

Variable	Expected Coriolis Acceleration	Simulated Coriolis Acceleration
a	$2\dot{\varepsilon_1}\dot{\varepsilon_2} = (2)(1 \text{ rad/s})(0.009901 \text{ rad/s})$ $= 0.019802 \text{ m/s}^2$	0.019465 m/s^2

Table 21: *The expected and simulated coriolis acceleration experienced by the second link of a* `SMS::BoomCrane`.

4.4.6 Conservation of Energy Test

4.4.6.1 Motivation

The purpose of this validation test is to ensure that the energy of the articulated body is always preserved. This is tested with a two-link mechanism where the proximal joint of the first link is a revolute joint with an initial non-zero joint rate, and the second link has a prismatic proximal joint with an initial joint rate of 0. The system is simulated until the joint rate of the first joint drops to zero because all of the energy was transferred to the second link. The energy at simulation start is compared against the energy at the end of the simulation.

4.4.6.2 Experimental Setup

The articulated body here is defined as in Section 4.4.4. To verify that energy is properly conserved, an initial velocity of 1 rad/s is applied to joint 1. A centripetal force will cause link 2 to actuate about its prismatic joint. The expected result is that link 1 will transfer all of its energy to link 2. The total kinetic energy at the start, $t = 0$, must be the same as the total energy at the end, $t = t_{el}$.

The energy balance equation can be described considering both the linear and angular components of energy as:

$$\frac{1}{2}m_{CG}^{(2)}(\dot{\varepsilon}_1(0)|\mathbf{P}_{1\to CG^{(2)}}|)^2 + \frac{1}{2}I_{zz}^{(2)}\frac{\dot{\varepsilon}_1(0)^2}{r} = \frac{1}{2}m_{(CG)}^{(2)}\dot{\varepsilon}_2(t_{el})^2 + \frac{1}{2}I_{zz}^{(2)}\frac{\dot{\varepsilon}_1(t_{el})^2}{r} \qquad (144)$$

4.4.6.3 Simulation Results

Table 22 compares the expected final velocity assuming kinetic energy was conserved against the simulated final velocity of the proximal joint of the 2nd link of the `SMS::BoomCrane`.

Variable	Expected Joint Velocity	Simulated Joint Velocity
$\dot{\varepsilon}_2$	1.1547 m/s	1.1565 m/s

Table 22: *The expected and final velocity of the proximal joint of the 2nd link of the* `SMS::Boomcrane` *conservation of energy test.*

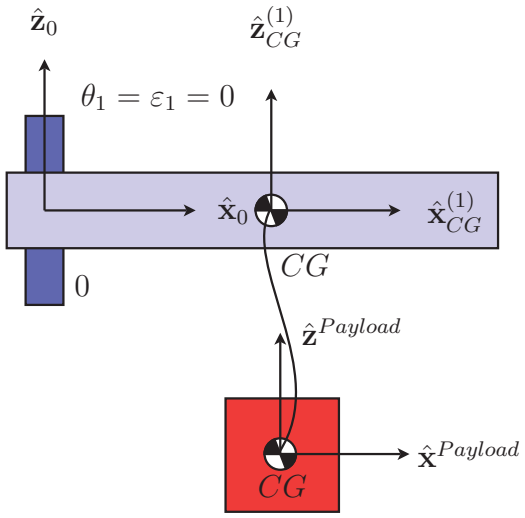


Figure 53: *The experimental setup for the boomcrane-cable-payload attachment validation test. Shown here with no joint displacement.*

4.4.7 BoomCrane-Cable-Payload Static Test

4.4.7.1 Motivation

The purpose of this validation test is to demonstrate that an `SMS::BoomCrane`, `SMS::Cable`, and `SMS::Payload` can all be attached together and will remain attached.

4.4.7.2 Experimental Setup

An articulated body is defined by the following D-H Parameters:

Link id	α_i	a_i	d_i	θ_i
1	0	0	0	0*

Table 23: *D-H Parameters for the payload attached to a cable that is attached to a boom crane steady state validation test.*

The base frame relative to the Earth-fixed frame, $\check{\mathbf{P}}_{E\to p}^{(1)}$ is made of the 3 Cartesian position components and 3 Euler angles that define its orientation. It is defined here

as $\check{\mathbf{P}}^{(1)}_{E\to p} = \begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T$.

The rigid body for link 1, a rectangular cuboid with a mass of 10 kg and dimensions of $L_x = 1$ m, $L_y = 0.025$ m, and $L_z = 0.025$ m has the following mass properties:

$$\mathbf{M}^{(1)}_{CG} = \begin{bmatrix} 10 & 0 & 0 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 & 0 & 0 \\ 0 & 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.01042 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3.334 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3.334 \end{bmatrix} \tag{145}$$

The position of the rigid body of the first link about the first joint, $\check{\mathbf{P}}^{(1)}_{p\to CG}$, made of the 3 Cartesian position components and 3 Euler angles that define the orientation, is defined as $\check{\mathbf{P}}^{(1)}_{p\to CG} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T$.

An axially stiff cable, made of two 10 m long elements, with three nodes at $[0, 0, 0]^T$, $[0, 0, -10]^T$, and $[0, 0, -20]^T$, is attached to the boom crane's first link, at $[1, 0, 0]^T$ with respect to the base. A distance is a scalar quantity.

A payload, whose rigid body is a rectangular cuboid with a mass of 5 kg, density of 7778 kg/m^3 and dimensions of $L_x = 0.01793$ m, $L_y = 0.01793$ m, and $L_z = 2$ m has the following mass properties:

$$\mathbf{M}_{CG} = \begin{bmatrix} 5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1.667 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.667 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2.679e{-}4 \end{bmatrix} \tag{146}$$

The other end of the cable is attached at $[0, 0, 1]^T$ with respect to the payload's body-fixed frame, CG, which is located at $[0, 0, -21]^T$.

The simulation is advanced for 30 seconds and the final payload position should be approximately $[0, 0, -21]^T$; there will be a small deviation due to cable elasticity.

4.4.7.3 Simulation Results

The final position of the `SMS::Payload` after the simulation time has elapsed is presented in Table 24 with the expected final position for comparison.

Variable	Expected Position	Simulated Position
$\mathbf{P}_{E\to CG}$	$\begin{bmatrix} 0 \\ 0 \\ -21 \end{bmatrix}$	$\begin{bmatrix} 0.00000 \\ 0.00000 \\ -21.00141 \end{bmatrix}$

Table 24: *The position of the* `SMS::Payload` *attached to an* `SMS::Boomcrane` *via an* `SMS::Cable` *after 30 seconds of simulation time.*

4.4.8 BoomCrane-Cable-Payload Centripetal Test

4.4.8.1 Motivation

The purpose of this validation test is to quantitatively show that an assembled `SMS::BoomCrane`, `SMS::Cable`, and `SMS::Payload` system will react in a predictable manner when the `SMS::Boomcrane`'s revolute joint is actuated. This is accomplished with a single link boom crane spinning in a circle about its revolute joint whose axis of rotation is aligned with $\hat{\mathbf{Z}}$. The centripetal acceleration of the payload will be resisted by the tension in the cable. The simulated cable tension is compared against analytical expectations.

4.4.8.2 Experimental Setup

An articulated body (see Figure 54) is defined by the following D-H Parameters:

Link id	α_i	a_i	d_i	θ_i
1	0	0	0	0^*

Table 25: *D-H Parameters for the payload attached to a cable that's attached to a boom crane centripetal force validation test.*

The base frame relative to the Earth-fixed frame of the boom crane is defined as: $\check{\mathbf{P}}^{(1)}_{E\to p} = \begin{bmatrix} -100 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}^T$ which is made of 3 Cartesian position components followed by 3 Euler angles which defines its orientation.

The rigid body for link 1, a rectangular cuboid with a mass of 67500 kg, density of 2700 kg/m^3 and dimensions of $L_x = 100$ m, $L_y = 0.5$ m, and $L_z = 0.5$ m has the following

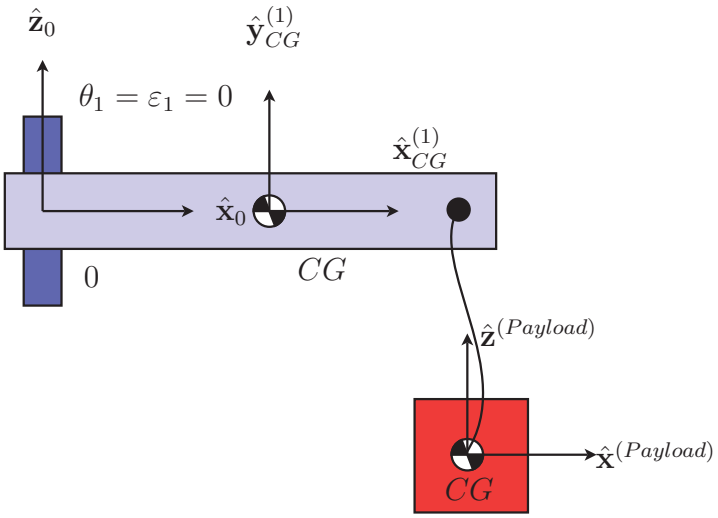


Figure 54: *The experimental setup for the boomcrane-cable-payload centripetal validation test shown with no joint displacement.*

mass properties:

$$\mathbf{M}_{CG}^{(1)} = \begin{bmatrix} 67500 & 0 & 0 & 0 & 0 & 0 \\ 0 & 67500 & 0 & 0 & 0 & 0 \\ 0 & 0 & 67500 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2812.5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 5.61e7 & 0 \\ 0 & 0 & 0 & 0 & 0 & 5.61e7 \end{bmatrix} \tag{147}$$

The position of the rigid body of the first link about the first joint, $\check{\mathbf{P}}_{p\to CG}^{(1)}$, made of the 3 Cartesian position components and 3 Euler angles that define its orientation, is defined as:

$$\check{\mathbf{P}}_{p\to CG}^{(1)} = \begin{bmatrix} 50 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{148}$$

An axially stiff cable, made of two 10 m long elements, with three nodes at $[0,0,0]^T$, $[0,0,-10]^T$, and $[0,0,-20]^T$, is attached to the boom crane's first link, at $[50,0,0]^T$ with respect to the CG of the link or at $[100,0,0]^T$ with respect to the proximal joint.

A payload, whose rigid body has a rectangular cuboid with a mass of 5 kg, density of 7778 kg/m^3 and dimensions of $L_x = 0.01793$ m, $L_y = 0.01793$ m, and $L_z = 2$ m has the following mass properties:

$$\mathbf{M}_{CG} = \begin{bmatrix} 5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1.667 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.667 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2.679e{-}4 \end{bmatrix} \tag{149}$$

The other end of the cable is attached at $[0,0,1]^T$ with respect to the payload's body-fixed frame, CG, which is located at $[0,0,-21]^T$.

A 350 second simulation was run. After 1 second of simulation time has elapsed, the boom crane's first joint is given a joint actuation rate of 0.35 rad/s. After 100 seconds, the tension in the cable at the payload attachment point is measured and the magnitude of the $\hat{\mathbf{X}}$ and $\hat{\mathbf{Y}}$ components is averaged over the last 10 seconds and compared against the expected centripetal force: $m_{payload}|\mathbf{P}_{p\to CG^{Payload}}^{(1)} - \mathbf{P}_{p\to CG^{Payload}}^{(1)} \cdot \hat{\mathbf{Z}}|(\dot{\varepsilon}_1(t_{el}))^2$.

4.4.8.3 Simulation Results

The resulting simulated force the `SMS::Cable` is applying on the `SMS::Payload` for this test is presented in Table 26 along with the expected analytical force for comparison. Note that the simulated force was sampled from the simulation when the `SMS::BoomCrane` link was rotating at a rate of 0.014712 rad/s.

Variable	Expected Horizontal Force	Simulated Horizontal Force
$\mathbf{F}(t_{el})$	$(5 \text{ kg})(100 \text{ m})(\dot{\varepsilon}_1(t_{el}))^2 = 0.106404$ N	0.105994 N

Table 26: *The simulated and expected centripetal force for the* `SMS::BoomCrane`-`SMS::Cable`-`SMS::Payload` *system.*

4.4.9 Crane Passive Joint Limit Test

4.4.9.1 Motivation

This validation test has two purposes: to show that a complex crane similar to a Palfinger-like mechanism can be modeled and to demonstrate that joint limits can be enforced.

4.4.9.2 Experimental Setup

The Palfinger crane (see Figure 55) has the following D-H Parameters[6]:

Link id	α_i	a_i	d_i	θ_i
1	90	-0.65	1.4	$0*$
2	0	1	0	0^*
3	90	0	0	90^*
4	0	0	0^*	0
5	0	0	0^*	0
6	0	0	0^*	0
7	0	0	0^*	0

Table 27: *D-H Parameters for the Palfinger joint validation test.*

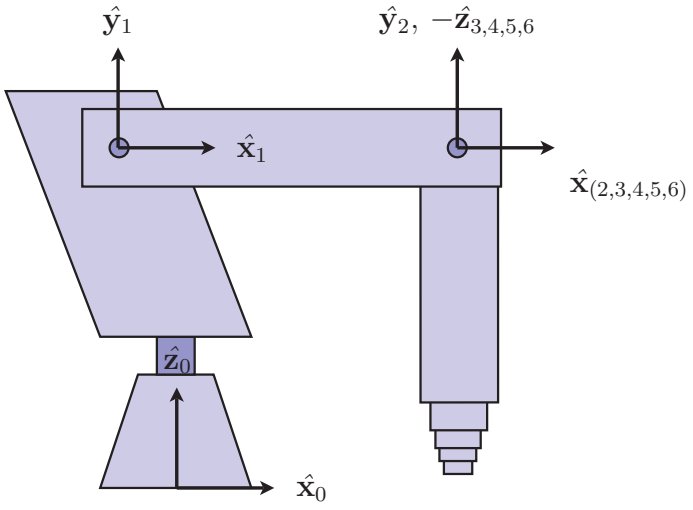


Figure 55: *The experimental setup for Palfinger boom crane validation tests. Shown here with no joint displacement.*

Its base frame relative to the Earth-fixed frame, $\check{\mathbf{P}}^{(1)}_{E\to p}$ is made of 3 Cartesian position

[6]These D-H parameters are not entirely representative of an actual Palfinger crane.

components followed by 3 Euler angles which defines its orientation, defined here as $\begin{bmatrix} -1.35 & 0 & -1.4 & 0 & 0 & 0 \end{bmatrix}$. This defines the first joint's frame.

The rigid body for link 1, a rectangular cuboid with a mass of 682.97 kg and dimensions of $L_x = 0.7$ m, $L_y = 0.7$ m, and $L_z = 1.4$ m has the following mass properties:

$$\mathbf{M}_{CG}^{(1)} = \begin{bmatrix} 682.97 & 0 & 0 & 0 & 0 & 0 \\ 0 & 682.97 & 0 & 0 & 0 & 0 \\ 0 & 0 & 682.97 & 0 & 0 & 0 \\ 0 & 0 & 0 & 139.44 & 0 & 0 \\ 0 & 0 & 0 & 0 & 139.44 & 0 \\ 0 & 0 & 0 & 0 & 0 & 55.776 \end{bmatrix} \tag{150}$$

and the rigid body for link 2, a rectangular cuboid with a mass of 754 kg and dimensions of $L_x = 1$ m, $L_y = 0.5$ m, and $L_z = 0.5$ m, has the following mass properties:

$$\mathbf{M}_{CG}^{(2)} = \begin{bmatrix} 754 & 0 & 0 & 0 & 0 & 0 \\ 0 & 754 & 0 & 0 & 0 & 0 \\ 0 & 0 & 754 & 0 & 0 & 0 \\ 0 & 0 & 0 & 31.42 & 0 & 0 \\ 0 & 0 & 0 & 0 & 78.5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 78.5 \end{bmatrix} \tag{151}$$

The rigid body for link 3, a rectangular cuboid with a mass of 285.22 kg and dimensions of $L_x = 1$ m, $L_y = 0.5$ m, and $L_z = 0.5$ m, has the following mass properties:

$$\mathbf{M}_{CG}^{(3)} = \begin{bmatrix} 285.22 & 0 & 0 & 0 & 0 & 0 \\ 0 & 285.22 & 0 & 0 & 0 & 0 \\ 0 & 0 & 285.22 & 0 & 0 & 0 \\ 0 & 0 & 0 & 11.8 & 0 & 0 \\ 0 & 0 & 0 & 0 & 29.71 & 0 \\ 0 & 0 & 0 & 0 & 0 & 29.71 \end{bmatrix} \tag{152}$$

The rigid body for link 4, a rectangular cuboid with a mass of 283.28 kg and dimensions of $L_x = 1$ m, $L_y = 0.5$ m, and $L_z = 0.5$ m, has the following mass properties:

$$\mathbf{M}_{CG}^{(4)} = \begin{bmatrix} 283.28 & 0 & 0 & 0 & 0 & 0 \\ 0 & 283.28 & 0 & 0 & 0 & 0 \\ 0 & 0 & 283.28 & 0 & 0 & 0 \\ 0 & 0 & 0 & 11.8 & 0 & 0 \\ 0 & 0 & 0 & 0 & 29.51 & 0 \\ 0 & 0 & 0 & 0 & 0 & 29.51 \end{bmatrix} \tag{153}$$

The rigid body for link 5, a rectangular cuboid with a mass of 254 kg and dimensions of $L_x = 1$ m, $L_y = 0.5$ m, and $L_z = 0.5$ m, has the following mass properties:

$$\mathbf{M}_{CG}^{(5)} = \begin{bmatrix} 254 & 0 & 0 & 0 & 0 & 0 \\ 0 & 254 & 0 & 0 & 0 & 0 \\ 0 & 0 & 254 & 0 & 0 & 0 \\ 0 & 0 & 0 & 10.58 & 0 & 0 \\ 0 & 0 & 0 & 0 & 26.46 & 0 \\ 0 & 0 & 0 & 0 & 0 & 26.46 \end{bmatrix} \tag{154}$$

The rigid body for link 6, a rectangular cuboid with a mass of 181.85 kg and dimensions of $L_x = 1$ m, $L_y = 0.5$ m, and $L_z = 0.5$ m, has the following mass properties:

$$\mathbf{M}_{CG}^{(6)} = \begin{bmatrix} 181.85 & 0 & 0 & 0 & 0 & 0 \\ 0 & 181.85 & 0 & 0 & 0 & 0 \\ 0 & 0 & 181.85 & 0 & 0 & 0 \\ 0 & 0 & 0 & 7.58 & 0 & 0 \\ 0 & 0 & 0 & 0 & 18.94 & 0 \\ 0 & 0 & 0 & 0 & 0 & 18.94 \end{bmatrix} \tag{155}$$

The rigid body for link 7, a rectangular cuboid with a mass of 95 kg and dimensions of $L_x = 1$ m, $L_y = 0.5$ m, and $L_z = 0.5$ m, has the following mass properties:

$$\mathbf{M}_{CG}^{(7)} = \begin{bmatrix} 95 & 0 & 0 & 0 & 0 & 0 \\ 0 & 95 & 0 & 0 & 0 & 0 \\ 0 & 0 & 95 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3.96 & 0 & 0 \\ 0 & 0 & 0 & 0 & 9.896 & 0 \\ 0 & 0 & 0 & 0 & 0 & 9.896 \end{bmatrix} \tag{156}$$

The position of the rigid bodies of each link about their proximal joint, $\check{\mathbf{P}}_{p\to CG}^{(i)}$ is made of the 3 Cartesian position components and 3 Euler angles that define their orientation and are defined as:

$$\check{\mathbf{P}}_{p\to CG}^{(1)} = \begin{bmatrix} -0.35 \\ 0.7 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{157}$$

$$\check{\mathbf{P}}^{(2)}_{p\to CG} = \begin{bmatrix} 0.5 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{158}$$

$$\check{\mathbf{P}}^{(3)}_{p\to CG} = \begin{bmatrix} 0 \\ -0.5 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{159}$$

$$\check{\mathbf{P}}^{(4)}_{p\to CG} = \begin{bmatrix} 0 \\ 0 \\ 0.5 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{160}$$

$$\check{\mathbf{P}}^{(5)}_{p\to CG} = \begin{bmatrix} 0 \\ 0 \\ 0.5 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{161}$$

$$\check{\mathbf{P}}^{(6)}_{p\to CG} = \begin{bmatrix} 0 \\ 0 \\ 0.5 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{162}$$

$$\check{\mathbf{P}}^{(7)}_{p\to CG} = \begin{bmatrix} 0 \\ 0 \\ 0.5 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{163}$$

Each joint of the crane has a PID *position* controller set to control their position at their initial position except for link 2's proximal joint, which is left *passive* (no PID Controller) and link 3's proximal joint is given an initial position of $\theta_3 = 90$. This will allow link 2's joint to try to deflect past its limit under gravity. Link 2's joint lower limit is set to $-45°$, with a joint stiffness of 10,000,000 Nm/rad and damping of 10,000,000 Nms/rad.

The final deflection of link 3's joint is compared against the joint limit ($-45°$) after an elapsed simulation time of 10 seconds.

4.4.9.3 Simulation results

Table 28 presents the expected and simulated final position of the Palfinger-like `SMS::BoomCrane`'s passive 2nd joint.

Variable	Expected Joint Position	Simulated Joint Position
ε_2	$-\frac{\pi}{4} = 0.7854$ rad	-0.7868 rad

Table 28: *The expected and simulated joint position for the Palfinger Crane Passive Joint Limit Test.*

See Figure 4.4.9.3 for the simulated position of every joint over time. Notice the position PID controllers held all joint positions except the passive joint which was stopped by the joint limit.

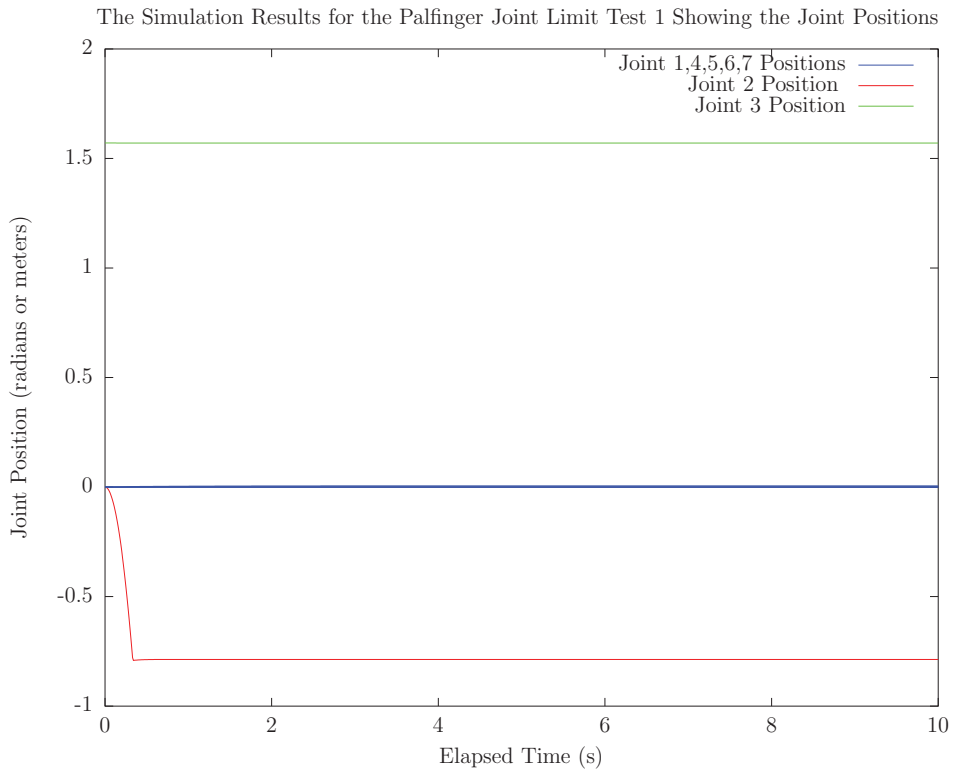


Figure 56: *The joint positions of the Palfinger for the passive joint limit violation test.*

4.4.10 Palfinger Crane Active Joint Limit Test

4.4.10.1 Motivation

The purpose of this validation test is to demonstrate the joint rate PID controller as well as demonstrate how the PID controllers will not violate joint limits.

4.4.10.2 Experimental Setup

The Palfinger crane as described in Section 4.4.9, has a PID *position* controller applied to link 2's joint, and it is set past its upper limit of 90°. The final joint position is compared against the upper limit after an elapsed simulation time of 10 seconds.

4.4.10.3 Simulation Results

The simulated and expected final joint position for this test is presented in Table 29.

Variable	Expected Joint Position	Simulated Joint Position
ε_2	$\frac{\pi}{2} = 1.5708$ rad	1.570791 rad

Table 29: *The simulated and expected joint position for the Palfinger Crane Active Joint Limit Test.*

See Figure 4.4.10.3 for the simulated position of every joint over time. Notice the position PID controllers held all joint positions except the velocity controlled joint which was stopped by the joint limit.

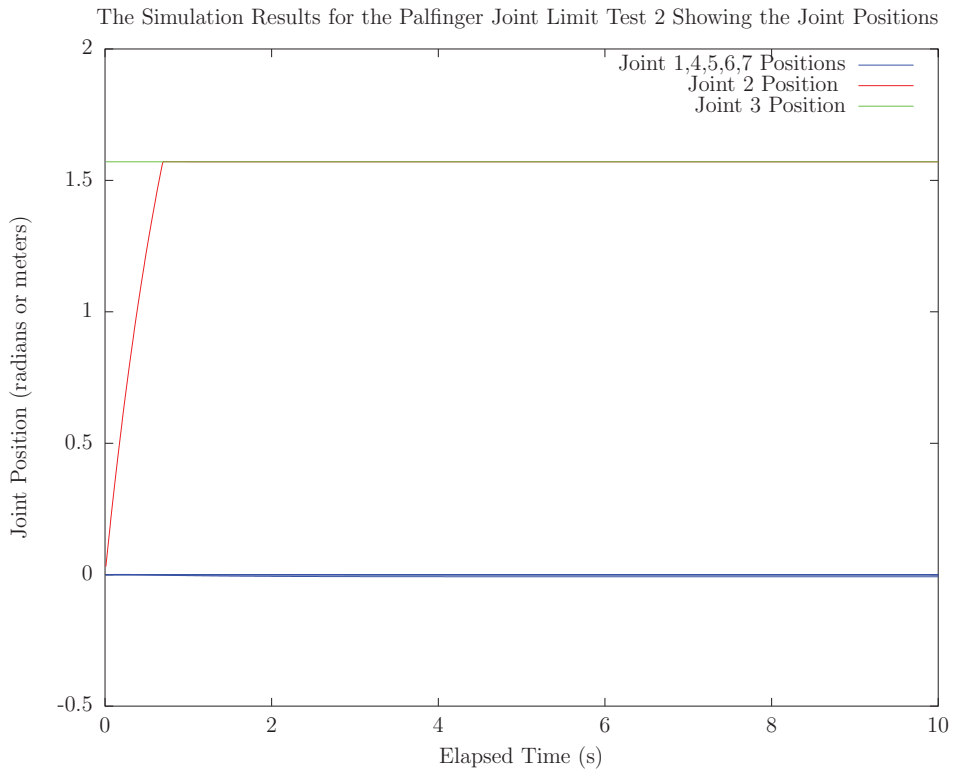


Figure 57: *The joint positions of the Palfinger for the active joint limit violation test.*

4.4.11 Palfinger Crane-Cable-Payload Dynamics Test 1

4.4.11.1 Motivation

The purpose of this validation test is to qualitatively show the behavior of an `SMS::Payload` when attached to an `SMS::Cable` whose other end is attached to an `SMS::Boomcrane` with its base translating at a constant velocity. The `SMS::Payload` has an initial velocity equal to the boom crane base velocity.

4.4.11.2 Experimental Setup

The Palfinger crane as described in Section 4.4.9, has link 2's joint PID *position* controller set at its initial position. The base of the Palfinger boom crane is given a constant velocity of $[1, 0, 0, 0, 0, 0]^T$.

A cable is attached to the Palfinger crane's link 7 rigid body at its CG and to a payload at a location $[0, 0, 1]$ with respect to its body-fixed frame. Both the cable and the payload have the properties described in 4.4.7.

The payload is given the same initial velocity as the Palfinger crane's base. The final position of the payload and boom crane after an elapsed period of time is compared against expected (noting that both their initial velocities are 1 m/s):

$$\check{\mathbf{P}}_{payload} = \begin{bmatrix} t_{elapsed} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{164}$$

$$\check{\mathbf{P}}^{(1)}_{p \to E} = \begin{bmatrix} t_{elapsed} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{165}$$

4.4.11.3 Simulation Results

The final velocity of the `SMS::Payload` is presented in Table 30 along with the expected analytical velocity.

Variable	Expected Payload Velocity	Simulated Payload Velocity
$\mathbf{v}_{pay}$	1 m/s	0.99163 m/s

Table 30: *The simulated and expected final velocity of the* `SMS::Payload` *for the Palfinger Crane-Cable-Payload Dynamics Test 1*

See Figure 4.4.11.3, which compares the position along $\hat{\mathbf{X}}$ of the base of the boom crane with payload's position along $\hat{\mathbf{X}}$.

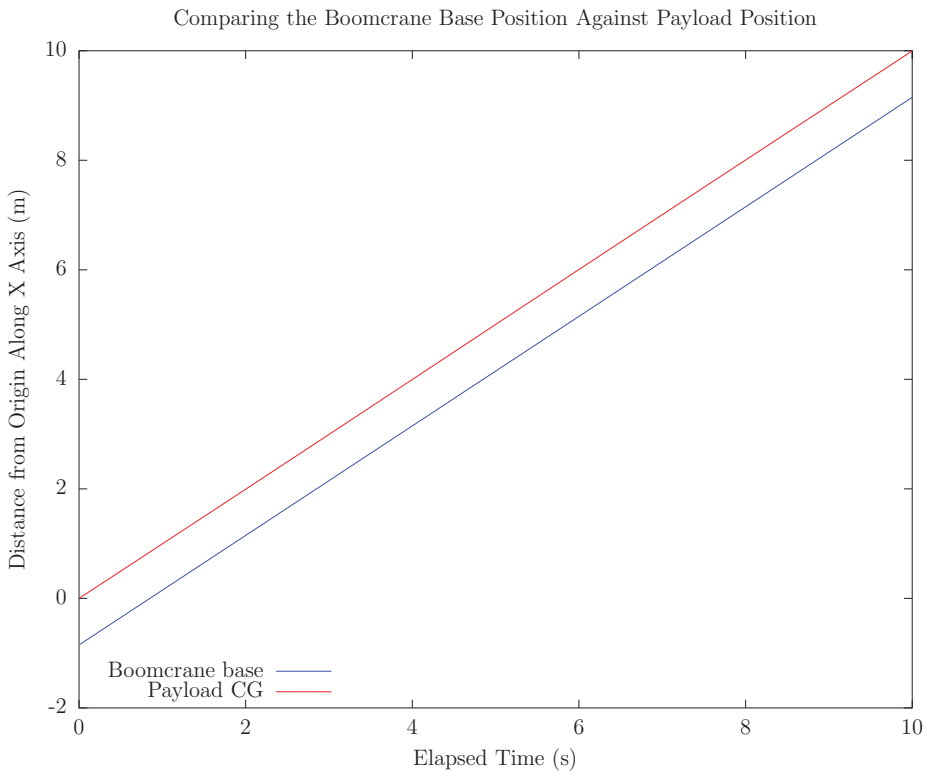


Figure 58: *The positions of both the boom crane's base and payload's CG over time, take note that the boom crane's base position does not start at origin.*

4.4.12 Palfinger Crane-Cable-Payload Dynamics Test 2

4.4.12.1 Motivation

The purpose of this validation test is to qualitatively show the behavior of the payload when attached to a cable whose other end is attached to a boom crane with a base moving at a constant velocity. The payload has no initial velocity.

4.4.12.2 Experimental Setup

The Palfinger crane as described in Section 4.4.9, has link 2's joint PID *position* controller set at its initial position. The base of the Palfinger boom crane is is given a velocity of $[1, 0, 0, 0, 0, 0]$.

A cable is attached to the Palfinger crane's link 7 rigid body at its CG and to a payload at a location $[0, 0, 1]$ with respect to its body-fixed frame. Both the `SMS::Cable` and the `SMS::Payload` have the properties described in 4.4.7.

The payload has no initial velocity. The payload is expected to oscillate about the cable's connection point on the Palfinger crane. The velocity of the `SMS::Payload` is compared against the velocity of the base of the `SMS::BoomCrane` after 80 seconds. The velocity of the payload is determined by the slope of a linear polynomial fitted across the position of the payload.

4.4.12.3 Simulation Results

The simulated final velocity of the `SMS::Payload` is presented in Table 31 along with the expected analytical velocity for comparison.

Variable	Expected Payload Velocity	Simulated Payload Velocity
$\lvert\mathbf{v}_{pay}\rvert$	1 m/s	1.002 m/s

Table 31: *The simulated and expected velocity of the* `SMS::Payload` *for the Palfinger Crane-Cable-Payload Dynamics Test 2*

See Figure 4.4.12.3, which compares the position along $\hat{\mathbf{X}}$ of the base of the boom crane with payload's position along $\hat{\mathbf{X}}$.

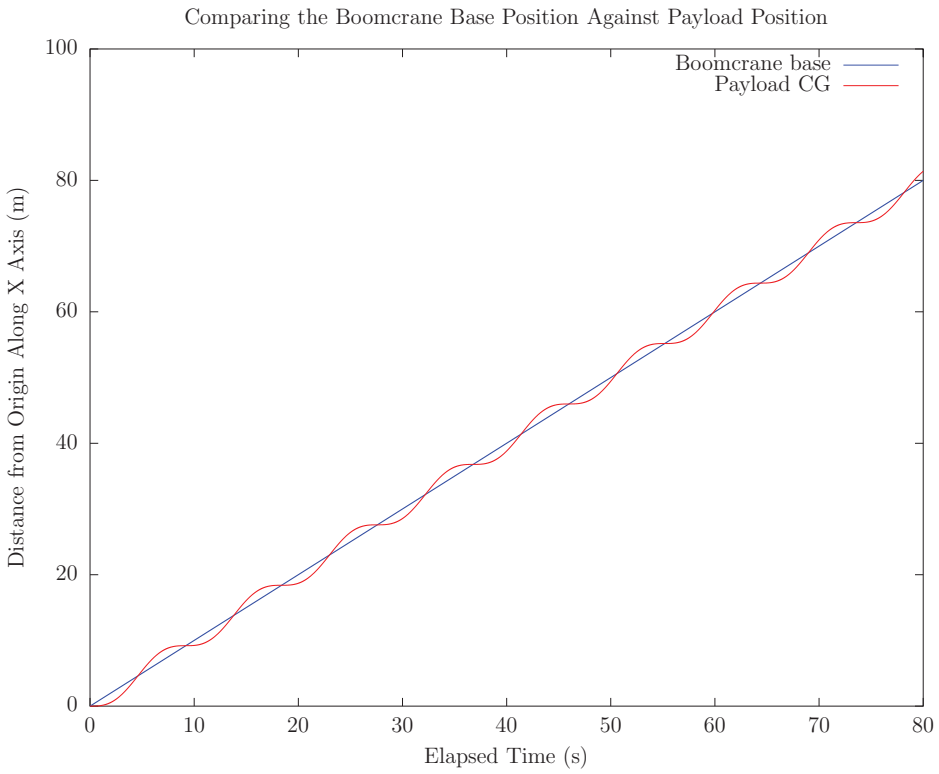


Figure 59: *The positions of both the boom crane's base and payload's CG over time*

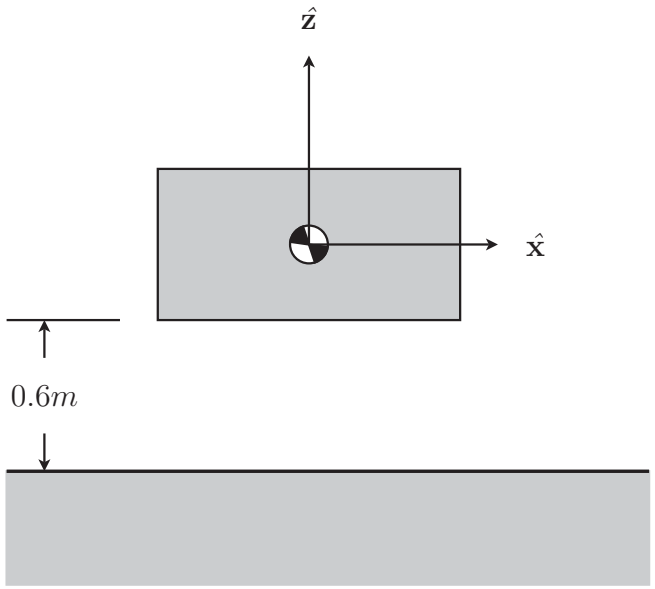


Figure 60: *The initial configuration of the box laying on ground test.*

4.5 Contact Resolution Validation

4.5.1 Box Laying Flat on Ground

4.5.1.1 Motivation

The purpose of this validation test is to qualitatively show that contacts between an `SMS::Payload` and an external object are resolved.

4.5.1.2 Setup

A box shaped `SMS::Payload`, as shown in Figure 60, whose bottom face normal is parallel with the $\hat{\mathbf{Z}}$ axis, is dropped under gravity to impact an external rigid body, another box, fixed in space, whose top face normal is also parallel with the $\hat{\mathbf{Z}}$ axis.

The external rigid body box has a length, width and height of 100 m, 100 m, and 10 m respectively and has its center of gravity located at $[0, 0, -6.1]$ placing its top face at -1.1 m along $\hat{\mathbf{Z}}$. The payload has dimensions of 5 m, 5 m, and 1 m, is originally located at $[0, 0, 0]$ placing its bottom face at -0.5 m along $\hat{\mathbf{Z}}$ and has a mass of 25,000 kg. The material of both the payload and the external rigid body are given a Young's modulus of 950 MPa and a damping coefficient of 50 MPa·s.

The simulation consists of allowing the `SMS::Payload` to accelerate along $-\hat{\mathbf{Z}}$ under gravity until the box makes contact with the external object and comes to rest on it

because of the energy losses of damped contact.

4.5.1.3 Simulation Results

At the beginning of the simulation, the box begins to accelerate in $-\hat{\mathbf{Z}}$ under gravity. When the box reaches a position of $\mathbf{P}_{E\to B} = [0, 0, -0.6, 0, 0, 0]^T$ an impact between the faces of both boxes occurs at $\mathbf{C}_\sigma = [0, 0, -1.1, 0, 0, 0]^T$. As shown in Figures 61 and 62, the box bounces off a few times each time losing energy from the material internal damping until it comes to a rest on the fixed external box.

Figure 63 shows the box maintains its original orientation, as is expected, with small high frequency self-correction oscillations.

There are large contact force spikes at each impact, as shown in Figure 64. They settle down and converge to equal the weight of the Payload: 245.25 kN. The position of the contact force remains fairly constant at the center of the contacting face with some high frequency, small amplitude, self-correcting oscillation about ${}^{E}\mathbf{P}_{contact} = [0, 0, -1.1]^T$. This can be seen in Figure 62.

The expected final volume of interference that would produce the steady state contact force equal to the weight, 2.4525e5 N, is obtained using the equivalent bed depth radius of 1.81 m and 28.79 m for the Payload and external box respectively and the material stiffnesses 950 MPa for both Payload and external boxes. Combining these values with equation 84, the expected volume of interference is 7.8996e-3 m^3. The equilibrium simulated volume of interference is differs by 0.03 percent, as seen in Figure 65 and Table 32.

Variable	Expected Volume of Interference	Simulated Volume of Interference
V_{int}	7.8996e-3 m^3	7.901706e-3 m^3

Table 32: *The simulated S.S. volume of interference for the Box Laying Flat on Ground Test*

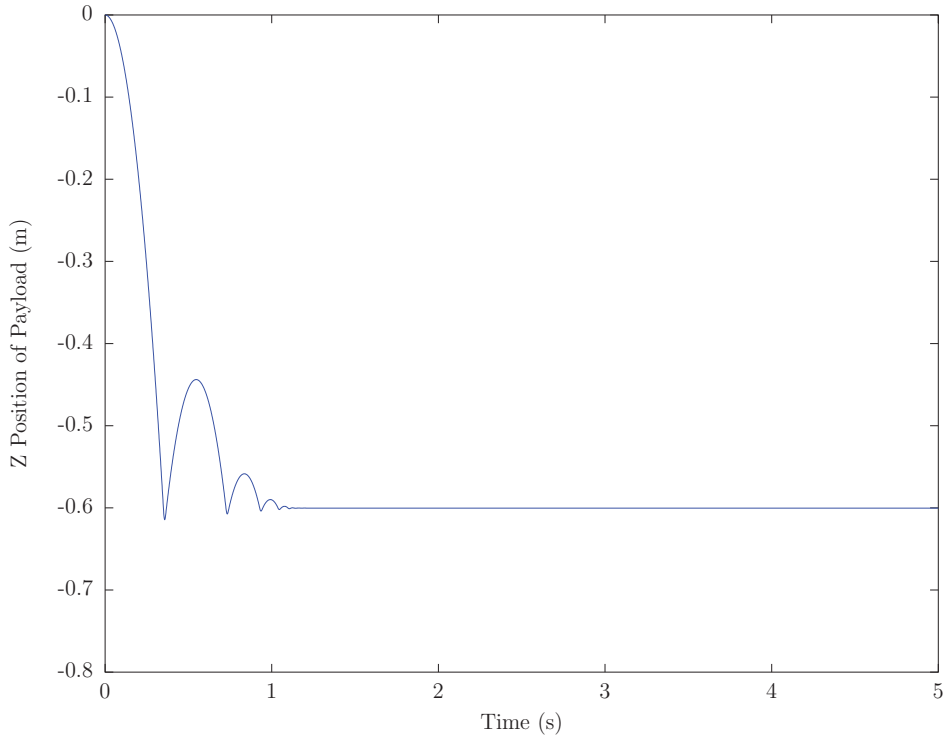


Figure 61: *The simulated position of a box impacting on the ground with a flat face.*

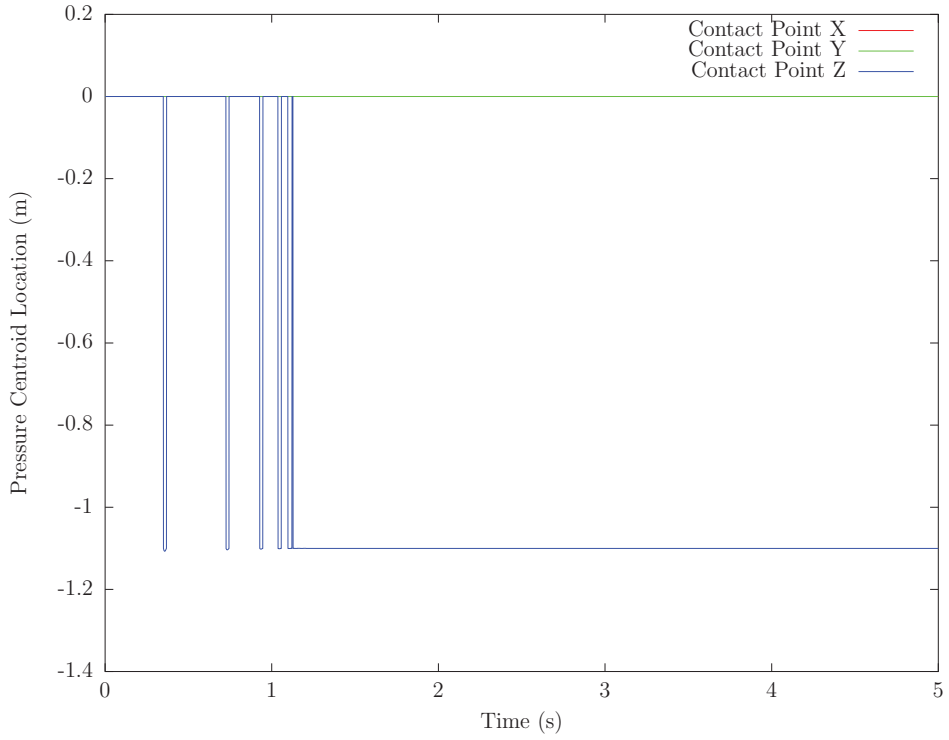


Figure 62: *The simulated contact point position of a box impacting on the ground with a flat face. Note that when there is no contact, the contact force location defaults to* $[0, 0, 0]^T$*, creating what appears to be large changes in* $\hat{\mathbf{Z}}$ *location of the contact point as the box comes into and leaves contact during the bouncing phase.*

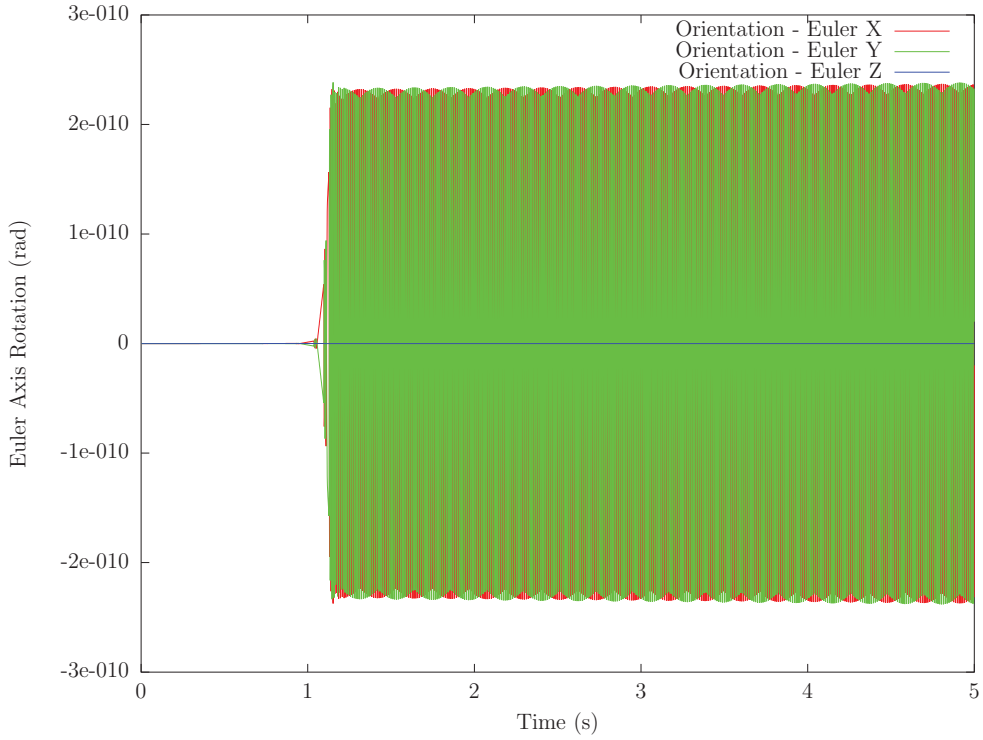


Figure 63: *The simulated orientation of a box impacting on the ground with a flat face.*

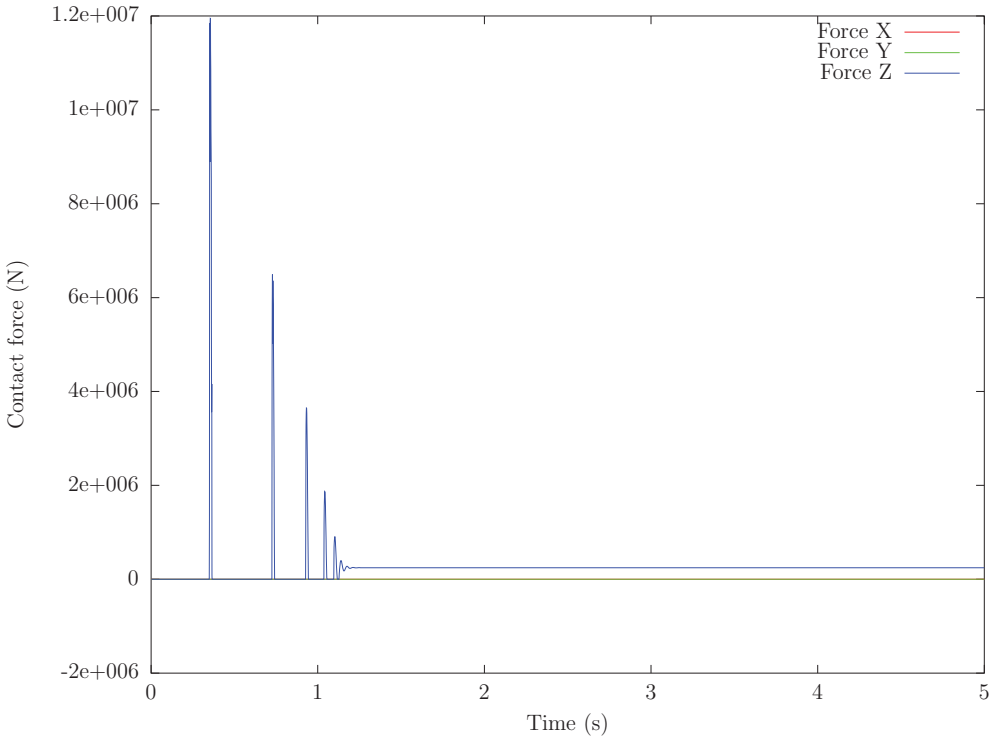


Figure 64: *The simulated contact forces of a box impacting on the ground with a flat face.*

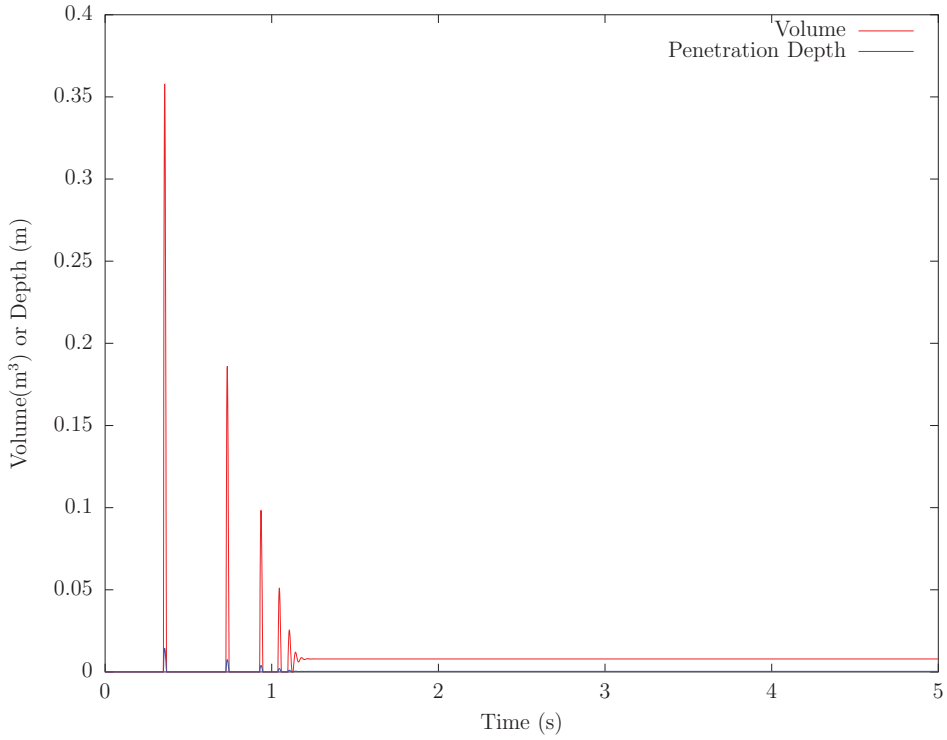


Figure 65: *The simulated volume of interference and penetration depth of a box impacting on the ground with a flat face.*

4.5.2 Oriented Box Impacting Ground

4.5.2.1 Motivation

The purpose of this test is to show how contact forces can be applied off-CG imparting moments on the `SMS::Payload`. In this case it will allow the `SMS::Payload` to come to rest on a flat surface, on one of its flat faces, even though it initially has no faces parallel to any faces of the external box.

4.5.2.2 Setup

A box shaped `SMS::Payload`, as shown in Figure 66, is given an initial position and orientation of $\check{\mathbf{P}}_{E\to B} = [0, 0, 0, 25.7°, 25.7°, 0°]^T$, is dropped to impact an external rigid body, another box, fixed in space, whose top face is horizontal.

The external rigid-body box has a length, width and height of 100 m, 100 m, and 10 m respectively and has its center of gravity located at $[0, 0, -15]$ placing its top face at -10 m. The payload has dimensions of 15 m, 15 m, and 5 m, is originally located at $[0, 0, 0]$ and a mass of 795 kg.

The material of both the payload and the external rigid body are given a Young's modulus of 100 MPa and a damping coefficient of 10 MPa·s.

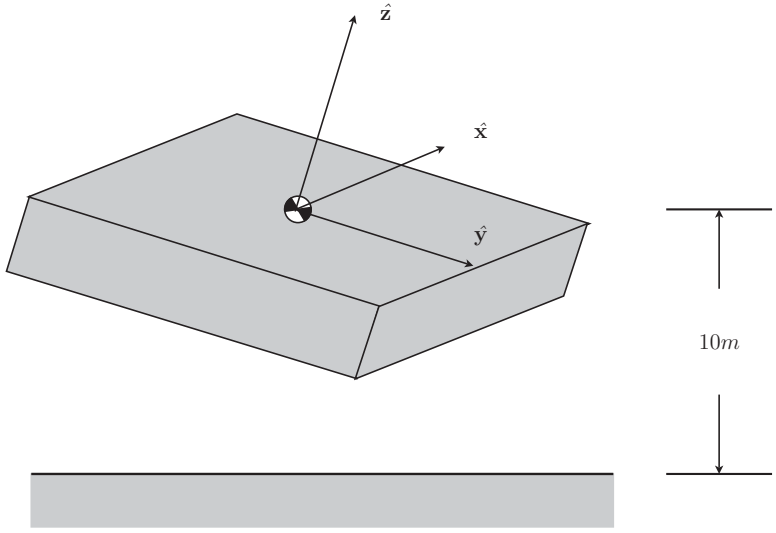


Figure 66: *The experimental setup of an oriented box impacting a flat surface.*

4.5.2.3 Simulation Results

At the beginning of the simulation, the box begins to accelerate along $-\hat{\mathbf{Z}}$ under gravity. When the box reaches a position of $\mathbf{P}_{E\to B} = [0, 0, -7.4974]^T$ an impact between the corner of the payload and the face of the external box at $\mathbf{C}_\sigma = [-3.9296, 6.9133, -10.003]^T$. The contact force vector is applied at the contact point and it doesn't pass through the center of gravity of the payload, causing rectifying moments. Eventually the material damps out the impacts until the `SMS::Payload` is laying flat on the external box object.

Figures 67 and 68 show the `SMS::Payload` impacts the external box bouncing and settles on one of its 6 faces. Here, the ϕ and θ Euler angles describing the orientation of the box settle at angles of 0, representing the bottom face, while the ψ angle slowly continues to rotate since there is no rotational friction.

There are large contact force spikes at each impact, seen in Figure 69, that settle down and converge to equal the weight of the `SMS::Payload`, 77989.5 N.

After the `SMS::Payload` settles, Figure 70 shows the position of the contact force oscillates.

The expected final volume of interference that would produce the steady state contact force equal to the weight, 245250 N, is obtained using the equivalent bed depth radius of 6.45 m and 28.79 m for the `SMS::Payload` and external box respectively and the material stiffnesses 10 MPa for both `SMS::Payload` and external boxes. Combining these values with equation 84, the expected volume of interference is 0.27484 m^3. The equilibrium simulated volume of interference differs by 1.84%, as seen in Figure 71 and Table 33.

Variable	Expected Volume of Interference	Simulated Volume of Interference
V_{int}	0.27484 m^3	0.28001 m^3

Table 33: *The simulated volume of interference for the Oriented Box Impacting Ground Test.*

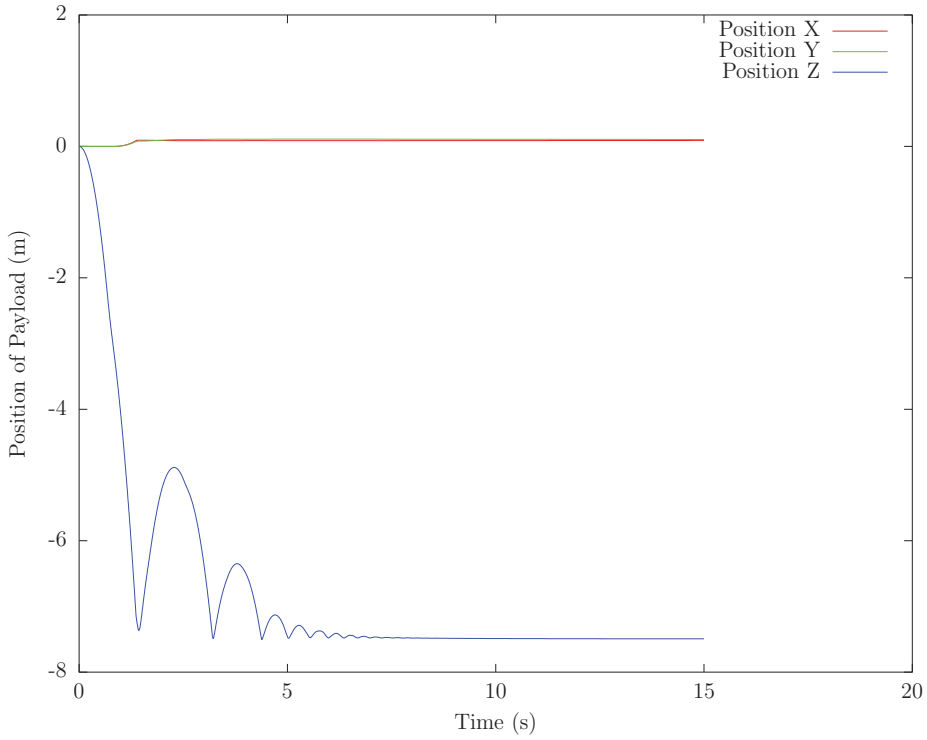


Figure 67: *The position of the payload's body-fixed frame.*

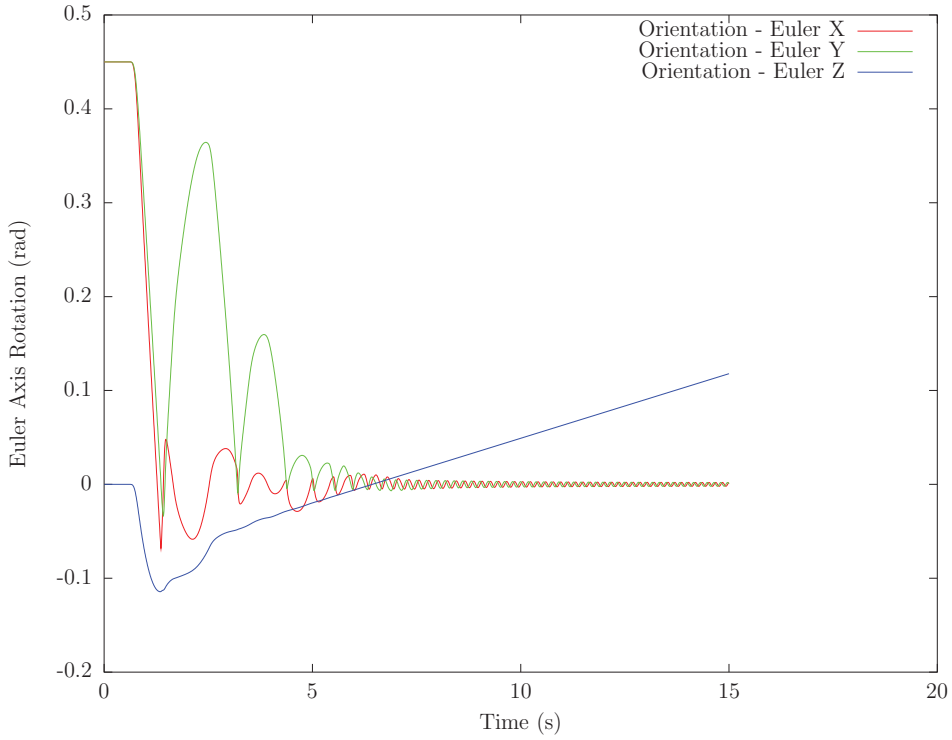


Figure 68: *The orientation of the payload's body-fixed frame.*

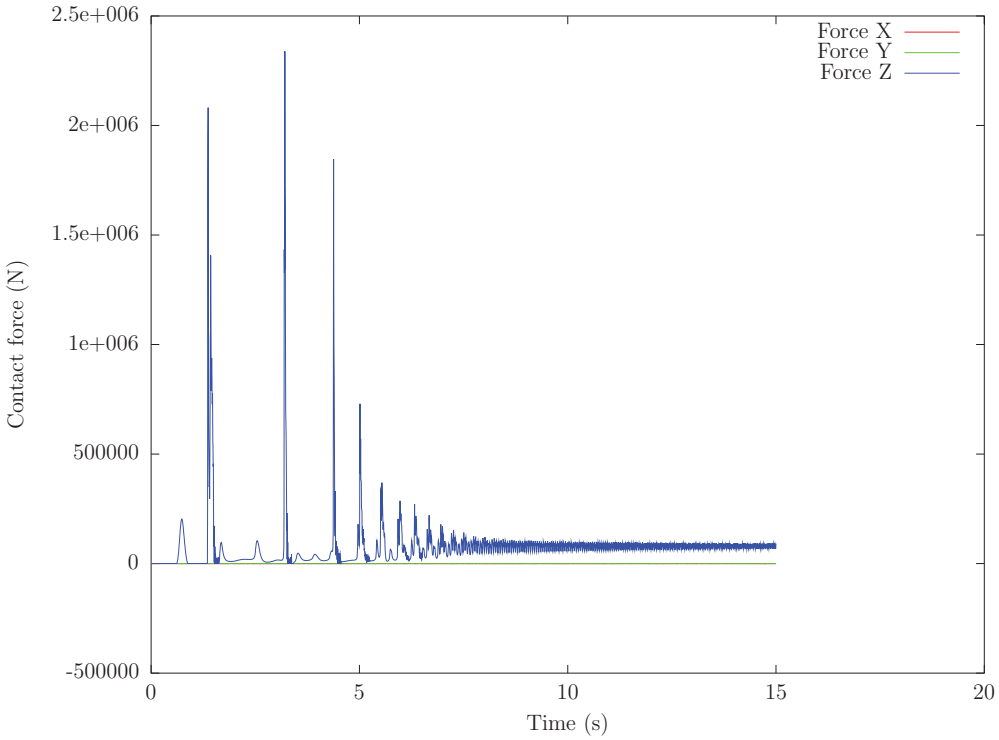


Figure 69: *The simulated contact forces experienced by the Payload for the oriented box test.*

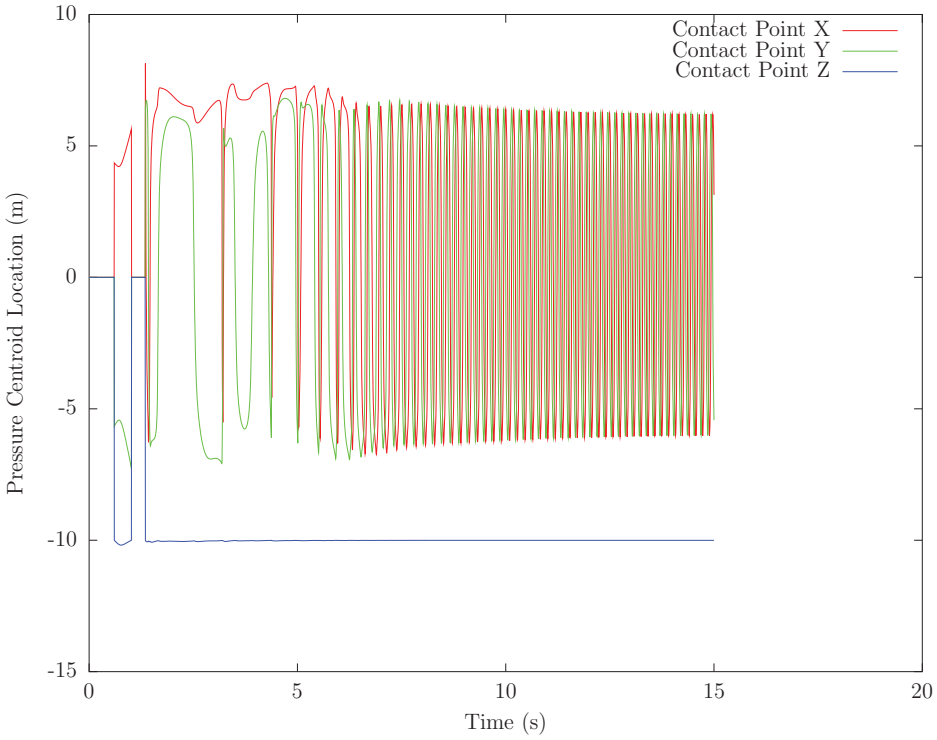


Figure 70: *The simulated contact point position of an oriented box impacting on the ground. Note that when there is no contact, the contact force location defaults to* $[0, 0, 0]^T$*, this creates what appears to be large changes in* $\hat{\mathbf{Z}}$ *location of the contact point as the box comes into and leaves contact during the bouncing phase. Also the large oscillations in the* $\hat{\mathbf{X}}$ *and* $\hat{\mathbf{Y}}$ *contact point position is caused by rapid shifting of the centroid of the volume of interference as the box makes small correction in its orientation (it's never laying perfectly flat).*

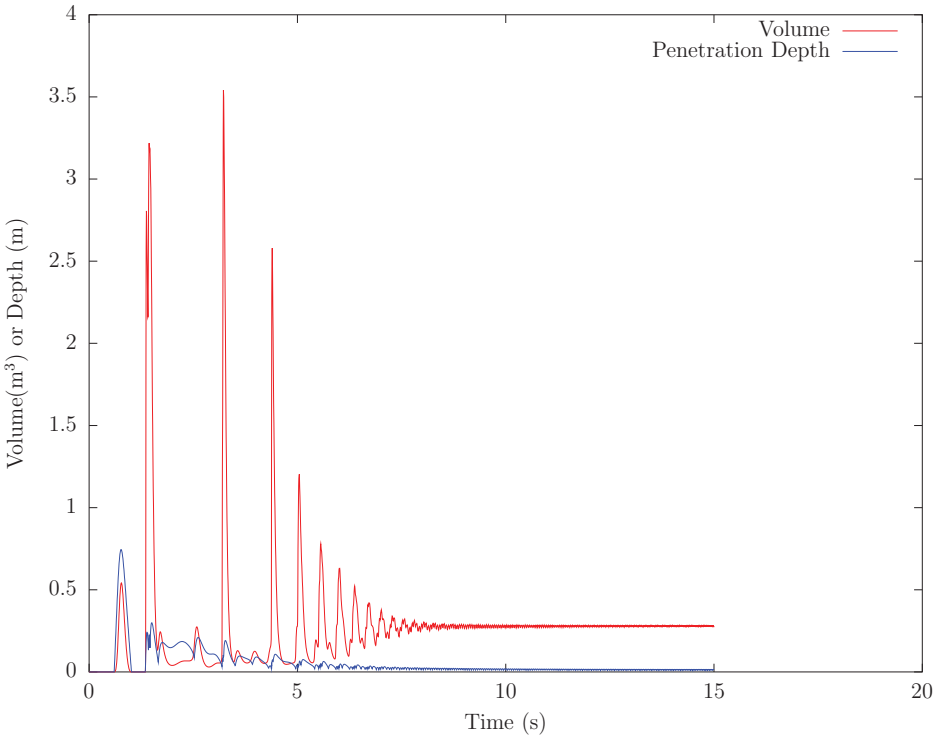
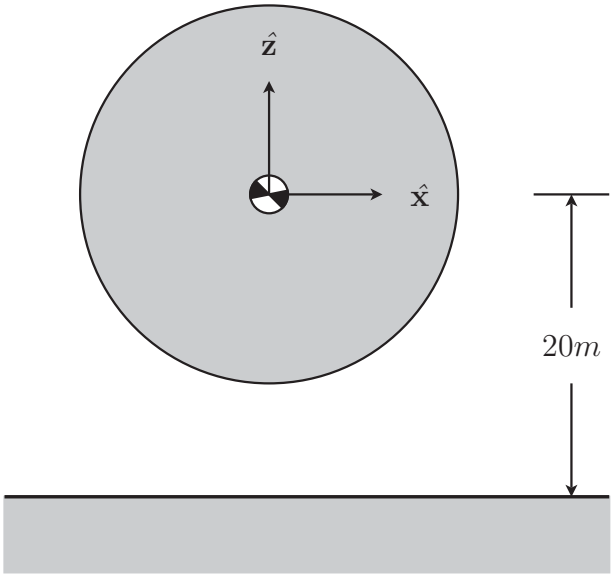


Figure 71: *The simulated volume of interference and penetration depth of a box impacting on the ground with a flat face.*



4.5.3 Conservation of Energy

4.5.3.1 Motivation

The purpose of this test is to show that energy is conserved during elastic impacts using volume of interference contact models.

4.5.3.2 Setup

A spherical `SMS::Payload`, with a radius of 1m, is given an initial position and orientation of $\check{\mathbf{P}}_{E\to B} = [0, 0, 0, 0°, 0°, 0°]^T$, and is dropped under gravity to impact an external rigid-body, a box, fixed in space, whose top face normal is parallel with the $\hat{\mathbf{Z}}$ axis.

The external rigid-body box has a length, width and height of 100 m, 100 m, and 10 m respectively and has its center of gravity located at $[0, 0, -25]$ placing it's top face at -20 m. The `SMS::Payload` has a radius of 1 m, and is originally located at $[0, 0, 0]$ with a mass of 3000 kg.

The material of both the `SMS::Payload` and the external rigid body are given a Young's modulus of 200 GPa and a damping coefficient of 1 Pa·s.

4.5.3.3 Simulation Results

At the beginning of the simulation, the spherical `SMS::Payload` begins to accelerate in $-\hat{\mathbf{Z}}$ under gravity. When the sphere reaches a position of $\mathbf{P}_{E\to B} = [0, 0, -19]^T$ an impact occurs between the sphere and the top face of the external box at $\mathbf{C}_\sigma = [0, 0, -20]^T$. This can be seen in Figure 72. Because there is practically no material damping there is no loss of energy during the contact. The initial impact velocity should equal final impact velocity and the `SMS::Payload` should bounce up to the same initial height continuously. Figure 72 demonstrates this behaviour.

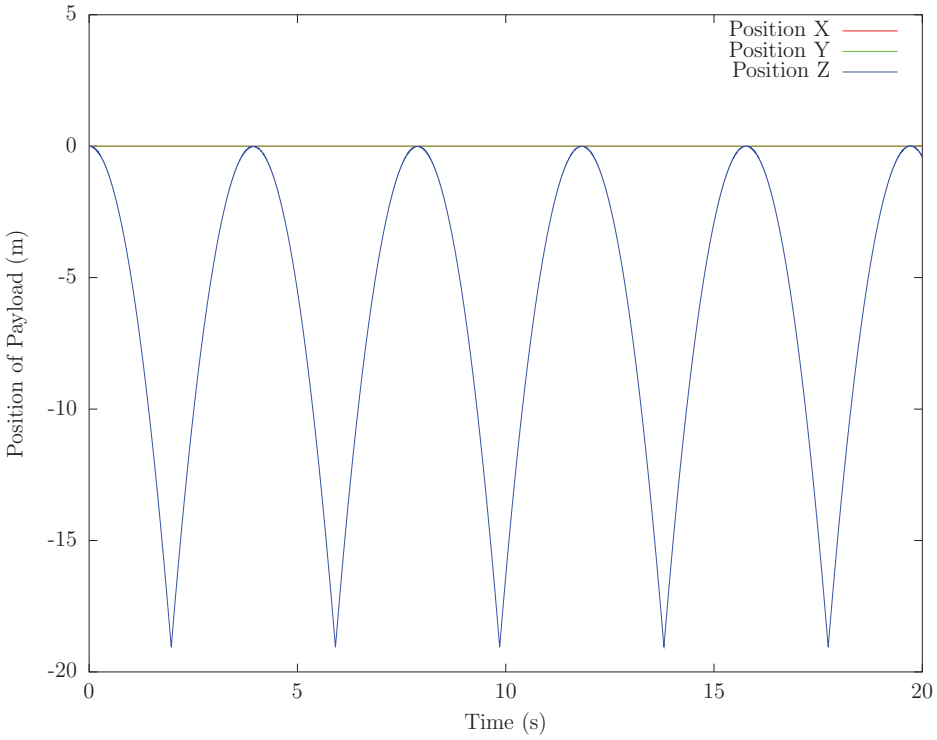


Figure 72: *The simulated position of a sphere, bouncing off a large fixed box, with no damping, showing a conservation of energy during the collision.*

4.5.4 All Sim Objects

4.5.4.1 Motivation

The purpose of this test is to qualitatively show that all simulation objects; `SMS::BoomCrane`, `SMS::Winch`, `SMS::Cable`, and a `SMS::Payload` can work in unison along with the contact resolution system.

4.5.4.2 Setup

A Palfinger-like crane, as shown in Figure 73, has the D-H Parameters shown in Table 34:

Link id	α_i	a_i	d_i	θ_i
1	90	-0.65	1.4	0∗
2	0	1	0	45*
3	90	0	0	90*
4	0	0	1*	0
5	0	0	1*	0
6	0	0	1*	0
7	0	0	1*	0

Table 34: *D-H Parameters for the Palfinger joint validation test.*

Its base frame relative to the Earth-fixed frame, $\check{\mathbf{P}}^{(1)}_{E\to p}$ is made of 3 Cartesian position components followed by 3 Euler angles which defines its orientation, defined here as $\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$. This defines the first joint's frame.

The rigid body for link 1, a rectangular cuboid with a mass of 682.97 kg and dimensions of $L_x = 0.7$ m, $L_y = 0.7$ m, and $L_z = 1.4$ m, has the following mass properties:

$$\mathbf{M}^{(1)}_{CG} = \begin{bmatrix} 682.97 & 0 & 0 & 0 & 0 & 0 \\ 0 & 682.97 & 0 & 0 & 0 & 0 \\ 0 & 0 & 682.97 & 0 & 0 & 0 \\ 0 & 0 & 0 & 139.44 & 0 & 0 \\ 0 & 0 & 0 & 0 & 139.44 & 0 \\ 0 & 0 & 0 & 0 & 0 & 55.776 \end{bmatrix} \tag{166}$$

and the rigid body for link 2, a rectangular cuboid with a mass of 754 kg and dimensions

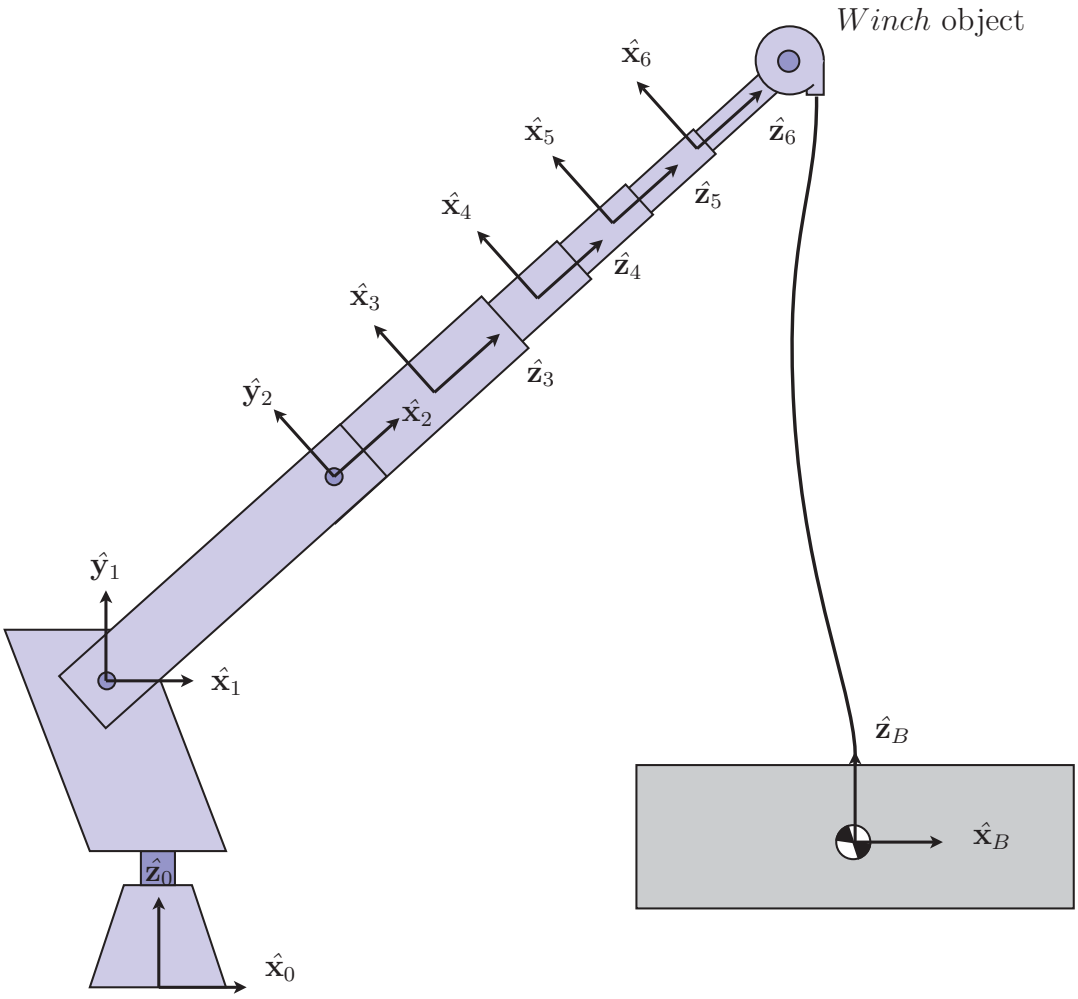


Figure 73: *The experimental setup for Palfinger boom crane validation tests. Shown here with no joint displacement.*

of $L_x = 1$ m, $L_y = 0.5$ m, and $L_z = 0.5$ m, has the following mass properties:

$$\mathbf{M}_{CG}^{(2)} = \begin{bmatrix} 754 & 0 & 0 & 0 & 0 & 0 \\ 0 & 754 & 0 & 0 & 0 & 0 \\ 0 & 0 & 754 & 0 & 0 & 0 \\ 0 & 0 & 0 & 31.42 & 0 & 0 \\ 0 & 0 & 0 & 0 & 78.5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 78.5 \end{bmatrix} \tag{167}$$

The rigid body for link 3, a rectangular cuboid with a mass of 285.22 kg and dimensions of $L_x = 1$ m, $L_y = 0.5$ m, and $L_z = 0.5$ m, has the following mass properties:

$$\mathbf{M}_{CG}^{(3)} = \begin{bmatrix} 285.22 & 0 & 0 & 0 & 0 & 0 \\ 0 & 285.22 & 0 & 0 & 0 & 0 \\ 0 & 0 & 285.22 & 0 & 0 & 0 \\ 0 & 0 & 0 & 11.8 & 0 & 0 \\ 0 & 0 & 0 & 0 & 29.71 & 0 \\ 0 & 0 & 0 & 0 & 0 & 29.71 \end{bmatrix} \tag{168}$$

The rigid body for link 4, a rectangular cuboid with a mass of 283.28 kg and dimensions of $L_x = 1$ m, $L_y = 0.5$ m, and $L_z = 0.5$ m, has the following mass properties:

$$\mathbf{M}_{CG}^{(4)} = \begin{bmatrix} 283.28 & 0 & 0 & 0 & 0 & 0 \\ 0 & 283.28 & 0 & 0 & 0 & 0 \\ 0 & 0 & 283.28 & 0 & 0 & 0 \\ 0 & 0 & 0 & 29.51 & 0 & 0 \\ 0 & 0 & 0 & 0 & 29.51 & 0 \\ 0 & 0 & 0 & 0 & 0 & 11.8 \end{bmatrix} \tag{169}$$

The rigid body for link 5, a rectangular cuboid with a mass of 254 kg and dimensions of $L_x = 1$ m, $L_y = 0.5$ m, and $L_z = 0.5$ m, has the following mass properties:

$$\mathbf{M}_{CG}^{(5)} = \begin{bmatrix} 254 & 0 & 0 & 0 & 0 & 0 \\ 0 & 254 & 0 & 0 & 0 & 0 \\ 0 & 0 & 254 & 0 & 0 & 0 \\ 0 & 0 & 0 & 26.46 & 0 & 0 \\ 0 & 0 & 0 & 0 & 26.46 & 0 \\ 0 & 0 & 0 & 0 & 0 & 10.58 \end{bmatrix} \tag{170}$$

The rigid body for link 6, a rectangular cuboid with a mass of 181.85 kg and dimensions

of $L_x = 1$ m, $L_y = 0.5$ m, and $L_z = 0.5$ m, has the following mass properties:

$$\mathbf{M}_{CG}^{(6)} = \begin{bmatrix} 181.85 & 0 & 0 & 0 & 0 & 0 \\ 0 & 181.85 & 0 & 0 & 0 & 0 \\ 0 & 0 & 181.85 & 0 & 0 & 0 \\ 0 & 0 & 0 & 18.94 & 0 & 0 \\ 0 & 0 & 0 & 0 & 18.94 & 0 \\ 0 & 0 & 0 & 0 & 0 & 7.58 \end{bmatrix} \tag{171}$$

The rigid body for link 7, a rectangular cuboid with a mass of 95 kg and dimensions of $L_x = 1$ m, $L_y = 0.5$ m, and $L_z = 0.5$ m, has the following mass properties:

$$\mathbf{M}_{CG}^{(7)} = \begin{bmatrix} 95 & 0 & 0 & 0 & 0 & 0 \\ 0 & 95 & 0 & 0 & 0 & 0 \\ 0 & 0 & 95 & 0 & 0 & 0 \\ 0 & 0 & 0 & 9.896 & 0 & 0 \\ 0 & 0 & 0 & 0 & 9.896 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3.96 \end{bmatrix} \tag{172}$$

The position of the rigid bodies of each link about their proximal joint, $\check{\mathbf{P}}_{p\to CG}^{(i)}$ is made of the 3 Cartesian position components and 3 Euler angles that define their orientation and are defined as:

$$\check{\mathbf{P}}_{p\to CG}^{(1)} = \begin{bmatrix} -0.35 \\ 0.7 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{173}$$

$$\check{\mathbf{P}}_{p\to CG}^{(2)} = \begin{bmatrix} 0.5 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{174}$$

$$\check{\mathbf{P}}_{p\to CG}^{(3)} = \begin{bmatrix} 0 \\ -0.5 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{175}$$

$$\check{\mathbf{P}}^{(4)}_{p\to CG} = \begin{bmatrix} 0 \\ 0 \\ 0.5 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{176}$$

$$\check{\mathbf{P}}^{(5)}_{p\to CG} = \begin{bmatrix} 0 \\ 0 \\ 0.5 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{177}$$

$$\check{\mathbf{P}}^{(6)}_{p\to CG} = \begin{bmatrix} 0 \\ 0 \\ 0.5 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{178}$$

$$\check{\mathbf{P}}^{(7)}_{p\to CG} = \begin{bmatrix} 0 \\ 0 \\ 0.5 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{179}$$

All joints of the Palfinger crane are given position joint controllers to maintain their initial joint positions.

An axially stiff `SMS::Cable` is attached to the Palfinger crane's link 7 rigid body at a location $[0, 0, 0.5]$ with respect to that link's body-fixed CG frame, located at $[3.592, 0, 5.642]$ and to a `SMS::Payload` at a location $[0, 0, 1]$ with respect to its body-fixed frame, located at $[3.592, 0, 0]$. The `SMS::Payload` is a 5 m $\times$ 5 m $\times$ 5 m box with a mass of 7800 kg. A `SMS::Winch` is attached to `SMS::Cable` at top 0 and is set to pay out 0.5 m of cable at a velocity of 0.1 m/s. The `SMS::Payload` will drop under gravity to impact an external rigid body, another box, fixed in space, whose top face normal is also parallel with the $\hat{\mathbf{Z}}$.

The external rigid-body box has a length, width and height of 100 m, 100 m, and 10 m respectively and has its center of gravity located at $[0, -7.6, 0.0]$ placing it's top face at -2.6 m along the $\hat{\mathbf{Y}}$ axis. The `SMS::Payload`'s initial location and orientation is $\check{\mathbf{P}}_{E\to B} = [3.592, 0, 0, 0, 0, 0]^t$

The material of both the `SMS::Payload` and the external rigid body are given a Young's modulus of 950 MPa and a damping coefficient of 50 MPa·s.

4.5.4.3 Simulation Results

At the beginning of the simulation, the `SMS::Winch` begins to pay out cable at a rate of 0.1 m/s until the box's position reaches $\mathbf{P}_{E\to B} = [0, 0, -0.5]^T$. Figure 74 shows that the `SMS::Payload` bounces off the external object. A contact occurs at $\mathbf{P}_{E\to \mathbf{C}_\sigma} = [3.636963, -3.991646e{-}4, -1.005504]^T$ as seen in Figure 75. The contact force is applied off of the center of gravity of the `SMS::Payload` causing moments. Eventually the material damps out the impact until the `SMS::Payload` lays flat on the external box object, in steady state.

Figure 76 shows the `SMS::Payload` impacts the external box, bouncing and settling on one of its 6 faces. Here the ϕ and θ Euler angles describing the orientation of the box settle at angles of 0. This means the bottom face of the box settled facing down, while the ψ angle slowly continues to rotate since there is no rotational friction.

There are large contact force spikes at each impact, as seen in Figure 77. They settle down and converge to equal the weight of the `SMS::Payload` $3.485958e4$ N, just about half of the weight of the `SMS::Payload`, $7.6518e4$ N.

After the `SMS::Payload` settles, the position of the contact force oscillates with the center of gravity of the volume of interference (affected by orientation) and leads to a rectifying moment, keeping the `SMS::Payload` sitting flat on the ground (see Figure 75).

Because a `SMS::Winch` is paying out cable, Figure 78 shows the `SMS::Cable`'s first node remains fixed in space, attached to the `SMS::Boomcrane` while the 2nd and 3rd nodes travel in $-\hat{\mathbf{Z}}$ until impact when the `SMS::Winch` is stopped. In Figure 79, the cable starts the simulation unloaded while the `SMS::Payload` drops under gravity, and settles at a tension equal to approximately half of the weight of the `SMS::Payload`.

The PID controller for the `SMS::Boomrane` holds the joints in their initial configuration effectively as can be seen in Figure 80. However, there are perturbations in the joints caused by the oscillations in the `SMS::Cable` tension. These can be seen in Figure 81 as the joint velocities oscillate about zero.

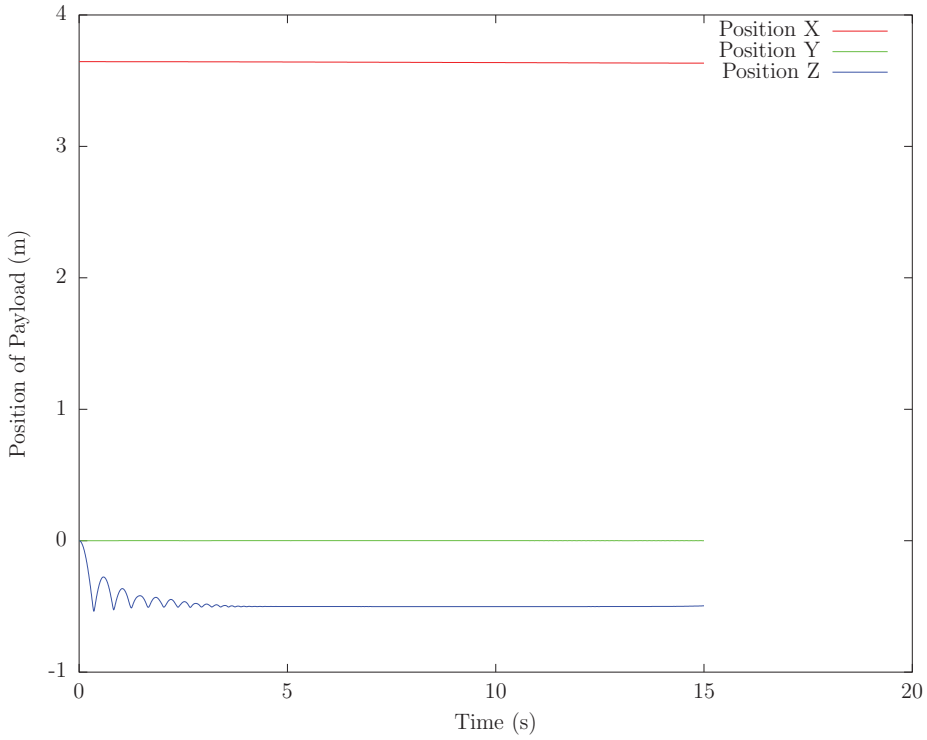


Figure 74: *The position of the payload's body-fixed frame.*

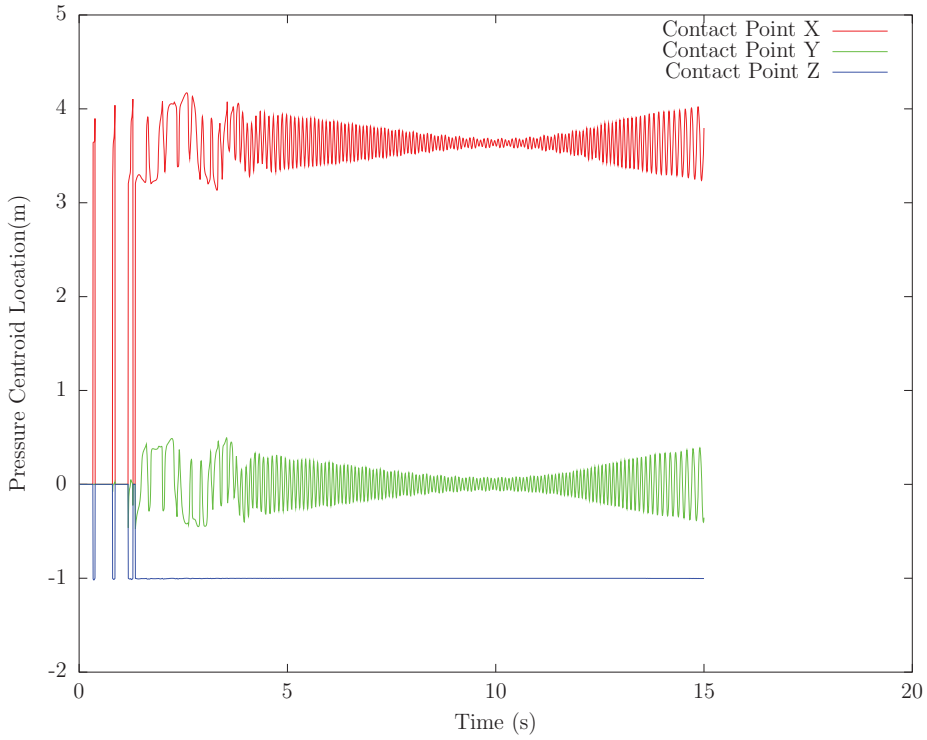


Figure 75: *The simulated position of a box impacting on the ground with a flat face.*

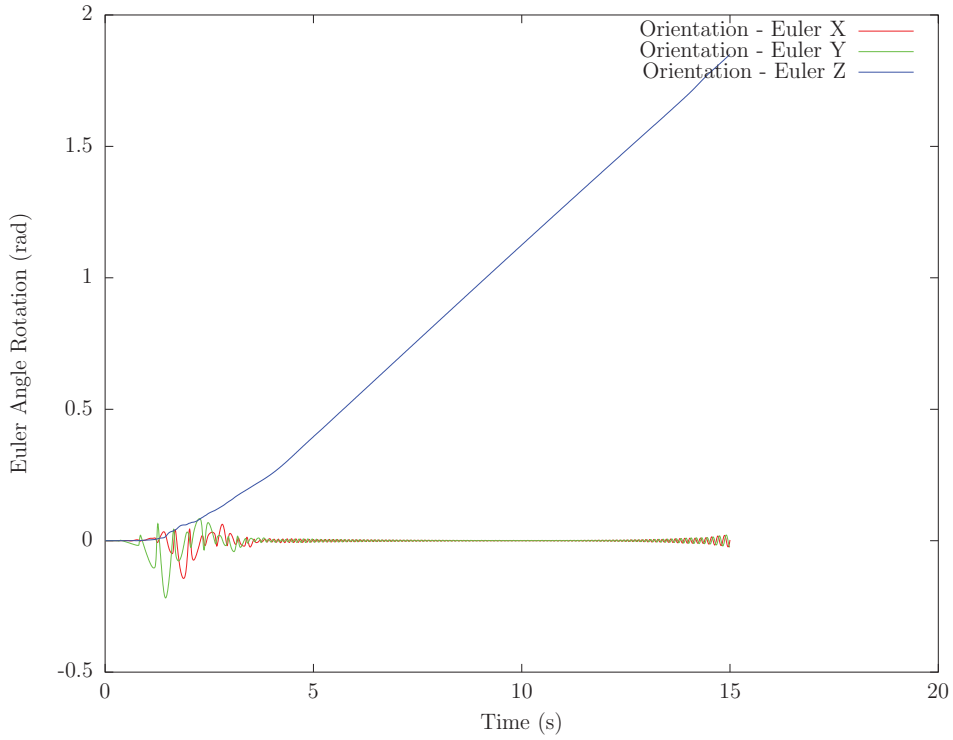


Figure 76: *The orientation of the payload's body-fixed frame.*

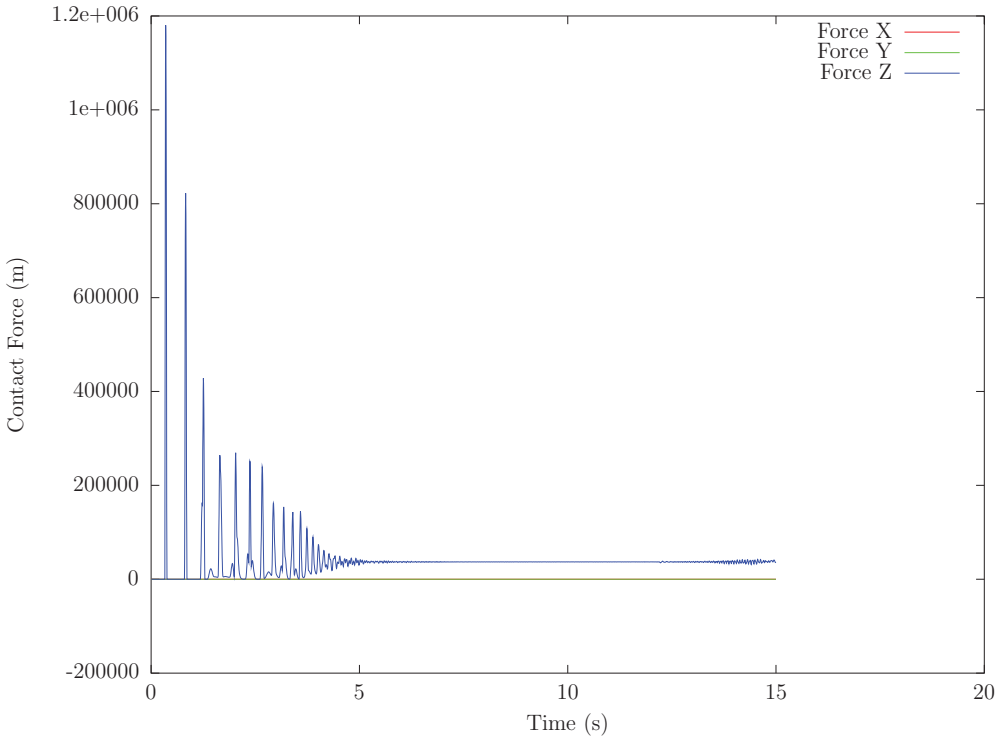


Figure 77: *The simulated contact forces experienced by the Payload for the oriented box test.*

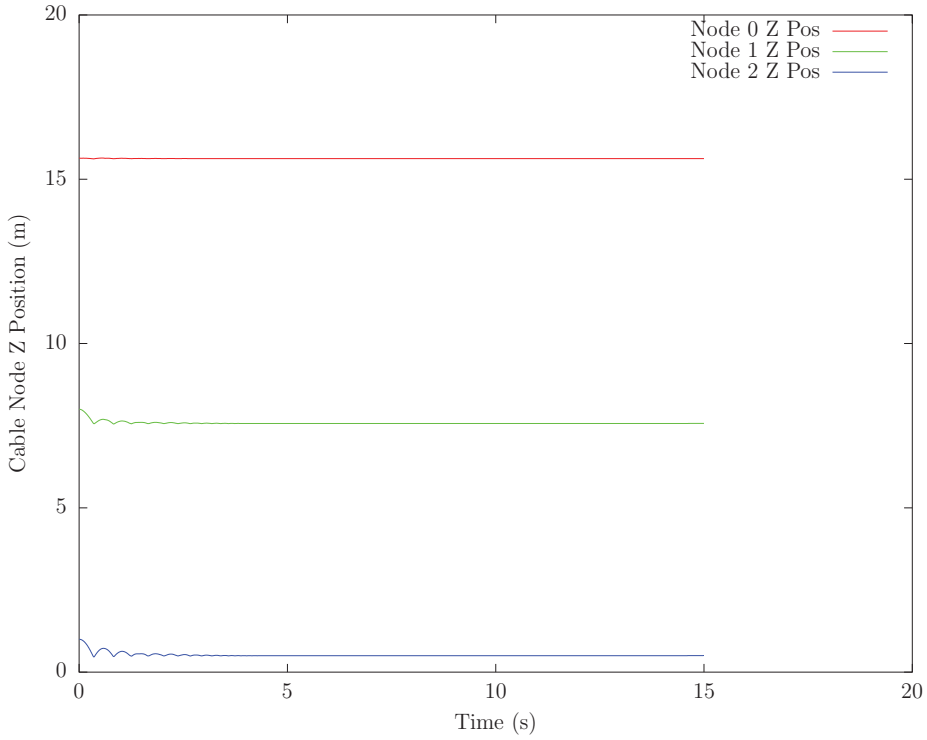


Figure 78: *The simulated cable node positions.*

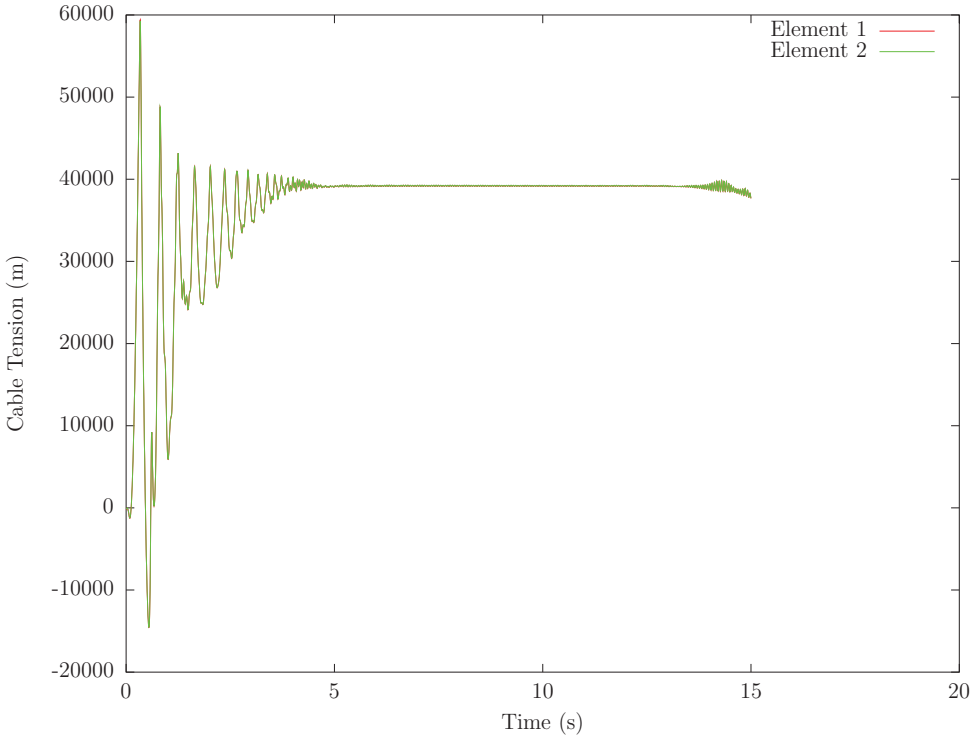


Figure 79: *The simulated cable element tensions.*

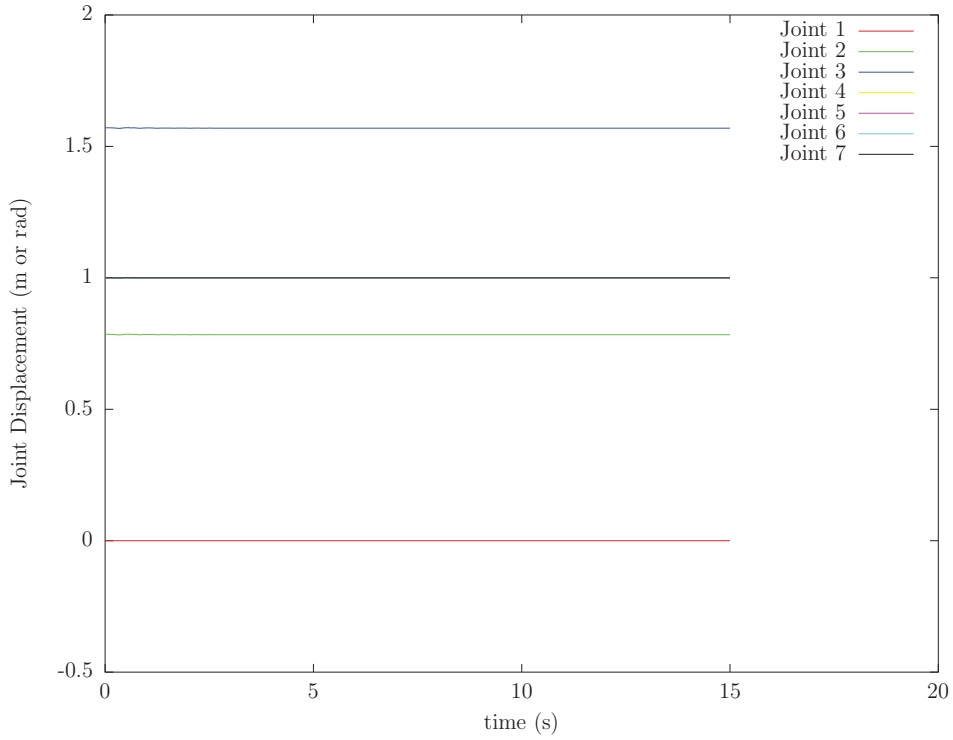


Figure 80: *The simulated boomcrane joint displacements.*

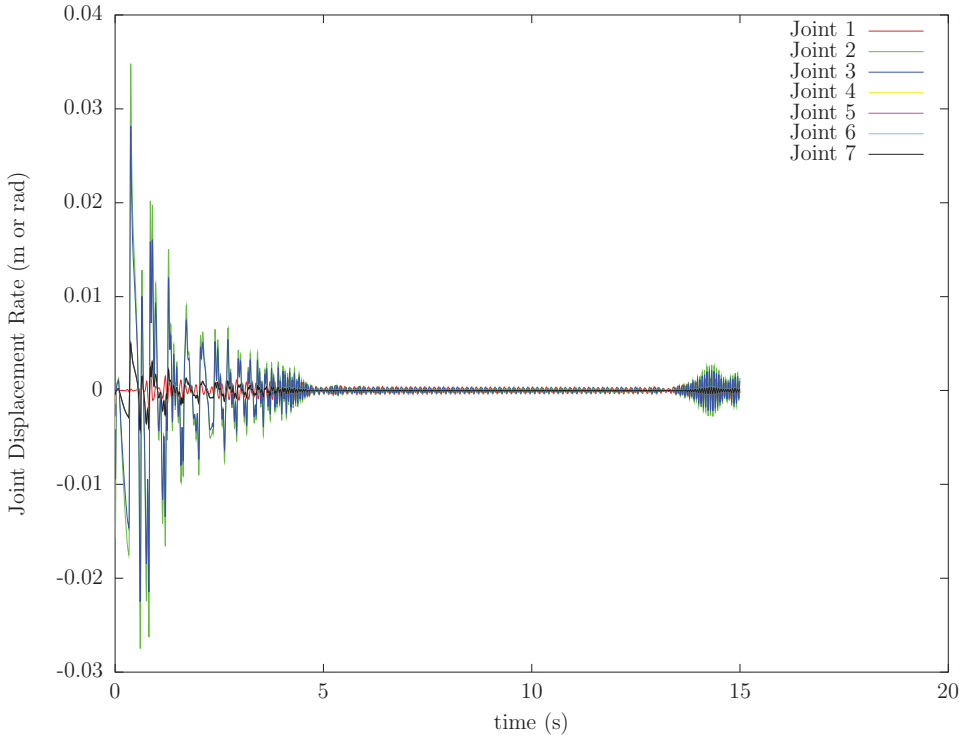


Figure 81: *The simulated boomcrane joint displacement rates.*

4.5.5 Potential Future validation tests

There are a number of other tests that could be run to improve the Contact Resolution validation. To demonstrate the sticking/sliding transition behaviour of the friction model, a treadmill friction test could be performed like that seen in Gonthier [29]. When spinning friction is accounted for in the friction model, a spinning top friction test could demonstrate the model's behaviour. The contact resolution system implemented in the SMS API can handle external objects with multiple convex features enabling multi-point contact. A test should be added to the validation suite to demonstrate the system functionality. The stiffness component of volume based contact models have been shown to adhere to Hertzian contact models [35]. The dynamic behaviour of the model should be compared against both Gonthier's model and empirical data.

5 Recommendations for Future Work

The classes implemented in the SMS API favor fidelity as much as practically possible. The purpose of the SMS API is to provide accurate engineering and analysis capabilities. Depending on the material properties used and the circumstances of the physical event modeled, simulations can often be executed in real-time. However, experience has shown that the SMS API will frequently run slower than real time, and will simulate significantly slower than real time during certain events (such as contact or cable snap loads). Since the SMS API presently uses an adaptive time step integrator, numerical integration time steps are reduced to prevent destabilization during shock events or with systems that have a high natural frequency. In particular, cable and contact dynamics are notoriously difficult systems to handle in that regard due to extremely high stiffnesses and usually small lumped masses (at the nodes of the cable mesh). As a result, many of the recommendations here focus on cable and contact dynamics modules. In general terms, the end goals of the recommendations are to:

- Increase the simulation execution speed through model adjustments and potentially implicit numerical integration;
- Profile and optimize the SMS API source code, removing "bottle necks" when possible;
- Add some functionality and features to the SMS API classes, such as clamped connections in the cable class and drum inertia in the winch class;
- Improve contact resolution execution speed by exploring the use of graphics card hardware and implementing a broad-phase collision detection scheme to handle multi-body contact simulations;
- Further advance the fidelity of the contact resolution models, and validate these enhancements.

It must be emphasized that a successful API is both maintainable (easy to read, update, and extend) and has good computational performance. It should be noted that computational performance will be a function of the efficiency of the data structures, code structure, and algorithms used, as well as the required time step to integrate the system physics. Identifying bottlenecks in simulation execution can sometimes only be done when a specific problem of interest has arisen and can be analyzed directly.

With that in mind, several specific improvements have been compiled in the following subsections that will potentially increase accuracy and computational performance of the methods and models described in this report. The benefits of each recommendation will vary depending on the expected use of the SMS API.

5.1 General Improvements

- Experience has shown the adaptive explicit numerical integrator used is robust and efficient for the physical systems modeled in the SMS API. However, new numerical integrators are often developed. A review of new and implicit numerical integration methods should be completed to check for alternatives. Hybrid explicit and implicit numerical integrators are also available and may allow larger time steps without destabilization.

- Preliminary code profiling of the SMS API has been completed. However, profiling has not been exhaustive and not all validation cases or any specific DRDC scenarios have been profiled. The SMS API code should be profiled to identify potential execution speed bottlenecks that can improve performance.

- No distributed simulation validation test cases have been written (simulations with SMS API objects in separate federates). Validation tests should be added to the validation test suite to ensure robustness. There are potential integration issues that may arise related to low frequencies of dynamic updates for boundary conditions of attached objects.

5.2 Cable Class Improvements

- The `SMS::Payload` and `SMS::Boomcrane` objects currently don't support clamped cable connections for separate federate simulations.

- For some sets of material properties and in particular with shorter element lengths, numerical time steps for integrating cable dynamics can be very small. Experience has shown that more accurate cable dynamics models usually lead to larger time steps that can be used. A modified consistent (un-lumped) mass model, based on the existing cable model, could be implemented that would more accurately model the local element rotational inertia effects and likely lead to time step increases. However, additional computational effort will be required due to larger systems of equations that must be solved for the same number of elements.

- When nodes are inserted or removed due to adaptive meshing (not currently implemented) or due to payin or payout operations, an artificial tension shock can occur. In order to minimize these effects, the use of an implicit integrator, such as Generalized-Alpha, that mitigates extreme high frequency effects while preserving low frequency dynamics, could be implemented. This can potentially reduce the need for potentially large structural damping values needed to produce larger time steps in explicit numerical integration methods.

- The Federate-based nature of the SMS API lends itself well to using different numerical integrators or numerical integration techniques for each dynamic model used. A novel hybrid explicit-implicit integration scheme could be investigated that uses a linear Generalized-Alpha implicit integrator during a pseudo-advance-time call to alleviate high frequency disturbances in between explicit numerical integration time steps. The linear Generalized-Alpha integrator would require specific knowledge of the cable profile and would be used for a small span of pseudo-time so as not to violate linearity assumptions, but act long enough to significantly reduce high frequency effects that result in small time steps with the explicit numerical integrator. This leverages the robust error control of the existing RK45 explicit numerical integrator and should significantly increase allowable simulation time step and therefore execution speed without any significant loss in accuracy.

5.3 Winch Class Improvements

- A relatively simple winch model that accounts for drum inertia and braking friction could be implemented to increase fidelity of the `SMS::Winch` class.

5.4 RigidBody and BoomCrane Class Improvements

- Adding support for quaternions in the rigid body-based models would avoid numerical stability problems that occur with Euler angles due to gymbal lock.

- The `SMS::Boomcrane` added mass hydrodynamic loading functionality is presently not implemented. This function should be implemented if articulated body simulations are required where hydrodynamic loading effects are significant.

5.5 Contact Resolution Improvements

- Graphics Processing Units (GPU) in off-the-shelf video cards, which contain vector-based processors rather than general-purpose processors, are being used more and more frequently to supplement computational capacity and improve simulation execution speed. A software library such as OpenCL or NVIDIA CUDA could be used to offload computational burdens from the CPU and would be particularly useful for hydrodynamics calculations or for geometrical algorithms used by the contact resolution system. In addition, since many of the cable dynamics calculations are serial, cables with longer elements could benefit from vector-based processing capabilities.

- Three dimensional contact stress effects, such as accounting for Poisson's ratio, would increase localized contact effect accuracy.
- The stiffness and damping terms of the contact model could be modeled through non-linear relationships between stress-strain and stress-strain rates respectively to increase the modelling accuracy. This would require introducing knowledge of the contact surface velocity, which is non-trivial.
- The contact model should be validated against experimental or empirical data and compared against other similar contact models.
- A more sophisticated friction model to help model lubricated conditions like wet ship decks can be implemented for increased accuracy (see the 3D LuGre model [26, 37]). Squeeze-film lubrication could also be taken into account.
- Rolling resistance and spinning friction should get implemented to increase accuracy of contact friction response.
- A more accurate normal contact force direction besides using the penetration depth, such as using polyhedral surface normal information, could increase accuracy.
- A better estimate of the contact bed depth should be found. Currently, an effective spherical radius is used, which is robust and efficient though crude.
- A better estimate of the centroid of pressure should be implemented. Currently the volume centroid is used as an approximation, though this approximation applies best for similar object materials.
- The MTD determination method is currently implemented as using a $O(N^2)$ method where N is the number of vertices of each colliding geometry. A more computationally efficient method exists that will allow for much better performance $O(logN)$ with more complex geometries and should be implemented to increase computational efficiency.
- The implementation of the data structures associated with the polyhedron mesh required for these efficient algorithms uses id-based structures. This makes deleting and creating vertices and polygons less efficient and limited in total number of vertex/edge/polygon additions. A pointer-based data structure would provide a more concise polyhedron object implementation.
- GJK has been implemented without exploiting temporal coherence, which can reduce the GJK algorithm to O(1) complexity. This would increase computational efficiency.

- In order to support multi-body contact situations, the SMS API contact resolution system must be modified and expanded to handle broad-phase collision detection. The `SMS::Boomcrane` and `SMS::Cable` classes could also benefit from inclusion in a broader contact resolution system for the SMS API. Currently, the `SMS::Payload` is the only object with contact resolution abilities and it can only model contacts with an external body (decomposed into convex geometrical features), which in turn is not influenced by the contact forces.

- The SMS API contact resolution system can currently handle concave objects by having the user manually decompose them into convex subobjects as a preprocessing step. An automatic convex decompositioning method or algorithms to handle concave objects should be implemented.

- The current SMS API contact resolution system must perform $O(N_A N_B)$ collision checks where N_A is the number of convex geometrical features that make up the `SMS::Payload` and N_B is the number of convex geometrical features that make up the external object.

- The current SMS API contact resolution system only checks for instantaneous collisions and does not account for tunneling. Tunneling is said to occur when two objects pass through each other in between time steps, and thus is non-detectable without the use of what is termed continuous collision detection. To handle tunneling, continuous collision detection methods would get implemented, such as through the use of swept volumes.

6 Conclusion

In conclusion, the model methodology and physical theory used in the development of the SMS API was presented. Four simulation object classes were discussed: the `SMS::Boomcrane` class, the `SMS::Winch` class, the `SMS::Cable` class, and the `SMS::Payload` class. A reference manual for the SMS API software was provided that describes the initialisation files for the different simulation objects as well as some of the important class methods. A validation test suite was developed to ensure the fidelity and accuracy of each of the simulation objects. The validation test suite also serves as a set of examples on how to use the SMS API to model various systems in different configurations. The motivation, experimental setup, and simulation results for each of the validation test suite cases were provided demonstrating the high fidelity simulation capabilities of the SMS API.

References

[1] Barber, C. B., Dobkin, D. P., and Huhdanpaa, H. (1996), The Quickhull Algorithm for Convex Hulls, *ACM Transactions on Mathematical Software*, 22 (4), 469–483.

[2] Ericson, E. (2005), Real-Time Collision Detection, The Morgan Kaufmann Series in Interactive 3D Technology, Morgan Kaufmann Publishers.

[3] Press, W. H. et al. (1988), Numerical Recipes in C, The Art of Scientific Computing, Cambridge University Press.

[4] Buckham, B. J. (2003), Dynamics modelling of low-tension tethers for submerged remotely operated vehicles, Ph.D. thesis, University of Victoria.

[5] Buckham, B., Nahon, M., and Cote, G. (2000), Validation of a finite element model for slack ROV tethers, In *OCEANS 2000 MTS/IEEE Conference and Exhibition*, Vol. 2, pp. 1129–1136 vol.2.

[6] Berstad, A. J., Tronstad, H., and Ytterland, A. (2004), Design rules for marine fish farms in Norway. Calculation of the structural response of such flexible structures to verify structural integrity, In *Proceedings of the International Conference on Offshore Mechanics and Arctic Engineering - OMAE*, Vol. 3, pp. 867–874, Vancouver, BC, Canada: American Society of Mechanical Engineers.

[7] O'Neill, B. (1966), Elementary Differential Geometry, Academic Press: New York, NY.

[8] Nahon, M. (1996), A Simplified Dynamics Model for Autonomous Underwater Vehicles, In *Proceedings of the 1996 IEEE Symposium on Autonomous Underwater Vehicle Technology*, pp. 373–379.

[9] Soylu, S. (2002), Incorporation of the articulated-body equations into a model-based sliding-mode controller for the reduction of dynamic coupling effect in underwater-manipulator systems, Master's thesis, University of Victoria.

[10] Ascher, U., Pai, D., and Cloutier, B. (December 1997), Forward Dynamics, Elimination Methods, and Formulation Stiffness in Robot Simulation, *International Journal of Robotics Research*, 16 (6), 749–758.

[11] Featherstone, R. (2008), Rigid body dynamics algorithms, Springer.

[12] McMillan, S., Orin, D. E., and McGhee, R. B. (1995), Efficient dynamic simulation of an underwater vehicle with a robotic manipulator, *IEEE Transactions on Systems, Man, and Cybernetics*, 25(8), 1194–1206.

[13] Craig, J. J. (1986), Introduction to Robotics: Mechanics and Control, Addison-Wesley Pub. Co., Reading Mass.

[14] McKerrow, P. J. (1991), Introduction to Robotics, Addison-Wesley Pub. Co., Reading Mass.

[15] Paul, R. and Shimano, B. (1981), Kinematic Control Equations for Simple Manipulators, *IEEE Transactions on Systems, Man and Cybernetics, SMC-11*, 6, 449–455.

[16] Lin, M. and Manocha, D. (2003), Collision and proximity queries, *In Handbook of Discrete and Computational Geometry, Chapman and Hall/CRC*.

[17] Jiménez, P., Thomas, F., and Torras, C. (2001), 3D Collision Detection: A Survey, *Computers and Graphics*, 25(2), 269–285.

[18] Gilardi, G. and Sharf, I. (2002), Literature Survey of Contact Dynamics Modelling, *Mechanism and Machine Theory*, 37(10), 1213–1239.

[19] Olsson, H., Åström, K. J., De Wit, C. C., Gafvert, M., and Lischinsky, P. (1998), Friction models and friction compensation, *European Journal of Control*, 4(3), 176–195.

[20] Carretero, J. A. and Nahon, M. A. (2005), Solving Minimum Distance Problems with Convex or Concave Bodies Using Combinatorial Global Optimization, *IEEE Transactions on Systems, Man and Cybernetics, Part B*, 35(6), 1144–1155.

[21] Uppuluri, R. and Carretero, J. A. (2004), A Novel Optimization-Based Pruning Strategy for Concave Minimum Distance Problems, In *Proceedings of the 2004 CSME Forum, London, Ontario*, pp. 581–591, London, Ontario, Canada.

[22] Mirtich, B. (1998), V-Clip: fast and robust polyhedral collision detection., *ACM Transactions on Graphics*, 17(3), 177–208.

[23] Gilbert, E. G., Johnson, D. W., and Keerthi, S. S. (1988), A Fast Procedure for Computing the Distance Between Complex Objects in three Dimentional Space, *IEEE Journal of Robotics and Automation*, 4(2), 193–203.

[24] Heidelberger, B., Teschner, M., Keiser, R., Mueller, M., and Gross, M. (2004), Consistent Penetration Depth Estimation for Deformable Collision Response, *Proceedings of Vision, Modeling, Visualization*, pp. 339–346.

[25] Luo, L. and Nahon, M. (2006), A Compliant Contact Model Including Interference Geometry for Polyhedral Objects, *ASME Journal of Computational and Nonlinear Dynamics*, 1(2), 150–159.

[26] Gonthier, Y., Mcphee, J., Lange, C., and Piedbœuf, J.-C. (2005), A Contact Modeling Method Based on Volumetric Properties, *Proceedings of IDETC'05 2005 ASME Design Engineering Technical Conferences and 5th International Conference on Multibody Systems, Nonlinear Dynamics and Control*.

[27] Ma, O. (1995), Contact Dynamics Modelling for the simulation of the Space Station Manipulators Handling Payloads, In *IEEE International Conference on Robotics and Automation*, pp. 1252–1258.

[28] Hasegawa, S. and Sato, M. (2004), Real-time Rigid Body Simulation for Haptic Interactions Based on Contact Volume of Polygonal Objects, *Eurographics*, 23(3).

[29] Gonthier, Y., McPhee, J., Lange, C., and Piedboeuf, J.-C. (2004), A Regularized Contact Model with Asymmetric Damping and Dwell-Time Dependent Friction, *Multibody System Dynamics*, 11, 209–233.

[30] Hippmann, G. (2003), An Algorithm For Compliant Contact Between Complexly Shaped Surfaces in Multibody Dynamics, In Ambrósio, J. A. C., (Ed.), *IDMEC/IST*.

[31] Johnson, K. L. (1987), Contact Mechanics, Cambridge University Press.

[32] Diolaiti, N., Melchiorri, C., and Stramigioli, S. (2005), Contact Impedance Estimation for Robotic Systems, *IEEE Transactions on Robotics*, 21(5), 925–935.

[33] Hunt, K. H. and Crossley, F. R. E. (1975), Coefficient of Restitution Interpreted as Damping in Vibroimpact, *Journal of Applied Mechanics*, 42, 440–445.

[34] Marhefka, D. W. and Orin, D. E. (1999), A Compliant Contact Model with Nonlinear Damping for Simulation of Robotic Systems, *IEEE Transactions on Systems, Man, and Cybernetics – PART A: Systems and Humans*, 29(6), 566–572.

[35] Roy, A. R. and Carretero, J. A. (2009), A Volume-Based Contact Dynamics Model with Asymmetric Damping Independent of the Coefficient of Restitution, In *Proceedings of the 2009 CCToMM Symposium on Mechanisms, Machines, and Mechatronics*.

[36] Goldsmith, W. (2001), Impact: The Theory and Physical Behaviour of Colliding Solids, Dover Publications.

[37] Canudas de Wit, C., Olsson, H., Åström, K. J., and Lischinsky, P. (1995), A New Model for Control of Systems with Friction, *IEEE Transactions on Automatic Control*, 40(3), 419–425.

[38] Kalpakjian, S. and Schmid, S. R. (2003), Manufacturing Processes for Engineering Materials, Prentice Hall.

[39] Muller, D. E. and Preparata, F. P. (1978), Finding the Intersection of Two Convect Polyhedra, *Theoretical Computer Science*, Vol. 7, 2.

[40] O'Rourke, J. (1998), Computational Geometry in C, 2nd ed, Cambridge University Press.

[41] Dobrovolskis, A. R. (1996), Inertia of any polyhedron, *Icarus*, 124, 698–704.

[42] Thomson, W. and Dahleh, M. D. (1998), Theory of Vibration With Applications, 5th edition, Prentice Hall.

[43] Zill, D. G. (1997), A first course in differential equations with modelling applications, sixth ed, Brooks Cole Publishing Company.

This page intentionally left blank.

Annex A: Consistent Element and Reduced Element Matrices

This appendix presents the full form of the consistent and lumped mass and stiffness matrices. In addition, a sample global stiffness matrix formed from two concatenated elements is presented. The consistent mass matrix is so-called because it was produced using the Galerkin criterion (Galerkin's method of weighted residuals) using the same shape functions that are used to quantify the discrete loads seen at the node points (see Appendix C). Later, in Equation 4, a lumped-mass approximation is made, leading to the use of the lumped-mass matrix that replaces the consistent mass matrix. For more details on how Galerkin's criterion is applied to the finite element cable model see [4].

The forms of the consistent stiffness matrices are described below. Referring to equation 2, the consistent discrete cable dynamics equation is:

$$\left(\mathbf{K}_{\epsilon} + \mathbf{K}_{\kappa} + \mathbf{K}_{\tau}\right)\mathbf{X} + \mathbf{W} + \mathbf{H} = \mathbf{M}_C\ddot{\mathbf{X}} \tag{A.1}$$

The 12×12 element consistent symmetric mass matrix is:

$$\mathbf{M}_C = L_u^{(1)} \begin{bmatrix} \frac{1}{3}\mathbf{M}_I^{(1)} & -\frac{L_u^{(1)}}{45}\mathbf{M}_I^{(1)} & \frac{1}{6}\mathbf{M}_I^{(1)} & -\frac{7(L_u^{(1)})^2}{360}\mathbf{M}_I^{(1)} \\ & \frac{2(L_u^{(1)})^4}{945}\mathbf{M}_I^{(1)} & -\frac{7(L_u^{(1)})^2}{360}\mathbf{M}_I^{(1)} & \frac{31(L_u^{(1)})^4}{15120}\mathbf{M}_I^{(1)} \\ & & \frac{1}{3}\mathbf{M}_I^{(1)} & -\frac{(L_u^{(1)})^2}{45}\mathbf{M}_I^{(1)} \\ SYM & & & -\frac{2(L_u^{(1)})^4}{945}\mathbf{M}_I^{(1)} \end{bmatrix} \tag{A.2}$$

where $L_u^{(1)}$ is the element unstretched length and $\mathbf{M}_I^{(1)}$ is the 3×3 element mass matrix comprised of the element physical and orientation-specific added mass:

$$\mathbf{M}_I^{(1)} = \frac{1}{4}\pi d_c^2 \rho_c \mathbf{I} + \frac{1}{2}\left(\mathbf{R}_{IH}^{(0)}\mathbf{M}_A^{(1)}\mathbf{R}_{IH}^{(0)T} + \mathbf{R}_{IH}^{(1)}\mathbf{M}_A^{(1)}\mathbf{R}_{IH}^{(1)T}\right) \tag{A.3}$$

where d_c is the cable diameter, ρ_c is the cable density, $\mathbf{I}$ is the 3×3 identity matrix, $\mathbf{R}_{IH}^{(i)}$ is the transformation matrix that relates the local hydrodynamic reference frame aligned to the tangent of the cable at node point i, to the inertial (global) coordinate frame, and $\mathbf{M}_A^{(1)}$ is the 3×3 local added mass matrix:

$$\mathbf{M}_A^{(1)} = \frac{1}{4}\pi d_c^2 \rho_w CAc \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{A.4}$$

Recall that the cable dynamics equation A.1 is formulated with respect to the global inertial reference frame. While added mass of the cable element is constant with respect to a reference frame aligned to the element, it must be transformed to be in terms of the global reference frame, which is why the transformation matrices are required. Note that since the hydrodynamic frames that are aligned to the tangent of the cable are resolved at the node points, the average of the resulting transformed added mass matrices evaluated at the element node points is used to provide a value of added mass of the element.

The lumped mass assumption concentrates all mass at the element end points and decouples curvature from all forces. The lumped mass matrix, $\mathbf{M}_G$, is formed by simply distributing half the element mass at either adjacent node:

$$\mathbf{M}_{G_{int}} = L_u^{(1)} \begin{bmatrix} \frac{1}{2}\mathbf{M}_I^{(1)} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ & & \frac{1}{2}\mathbf{M}_I^{(1)} & \mathbf{0} \\ SYM & & & \mathbf{0} \end{bmatrix} \tag{A.5}$$

Note $\mathbf{0}$ indicates a 3×3 zero matrix. After removing rows related to curvature acceleration the final 6×6 lumped mass matrix is:

$$\mathbf{M}_G = L_u^{(1)} \begin{bmatrix} \frac{1}{2}\mathbf{M}_I^{(1)} & \mathbf{0} \\ \mathbf{0} & \frac{1}{2}\mathbf{M}_I^{(1)} \end{bmatrix} \tag{A.6}$$

The lumped mass matrices must be concatenated in order to form the global system of equations that governs a system with N nodes. The global lumped mass matrix, $M_{G_{sys}}$ for a system with $N = 3$ nodes is:

$$\mathbf{M}_{G_{sys}} = \begin{bmatrix} \frac{1}{2}L_u^{(1)}\mathbf{M}_I^{(1)} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \frac{1}{2}\left(L_u^{(1)}\mathbf{M}_I^{(1)} + L_u^{(2)}\mathbf{M}_I^{(2)}\right) & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \frac{1}{2}L_u^{(2)}\mathbf{M}_I^{(2)} \end{bmatrix} \tag{A.7}$$

Note in equation A.7 that the global system mass matrix, which is formed using the lumped mass matrix in equation A.7, is block-diagonal with block size 3×3. If the global

mass matrix was assembled using the consistent mass matrix seen in Equation A.2, the system would be block-diagonal with block size 12×12. Since the computational time to resolve the system of equations grows nonlinearly with the size of the matrix, the lumped system of equations offers significant computational savings.

The element consistent axial stiffness matrix is:

$$\mathbf{K}_\epsilon = \frac{EA\epsilon^{(1)}}{L_u^{(1)}} \begin{bmatrix} -\mathbf{I} & \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & -\frac{1}{45}(L_u^{(1)})^4\mathbf{I} & \mathbf{0} & -\frac{7}{360}(L_u^{(1)})^4\mathbf{I} \\ \mathbf{I} & \mathbf{0} & -\mathbf{I} & \mathbf{0} \\ \mathbf{0} & -\frac{7}{360}(L_u^{(1)})^4\mathbf{I} & \mathbf{0} & -\frac{1}{45}(L_u^{(1)})^4\mathbf{I} \end{bmatrix} \tag{A.8}$$

where $\epsilon^{(1)}$ and $L_u^{(1)}$ are the first element strain and unstretched lengths, respectively. The first element consistent bending stiffness matrix is:

$$\mathbf{K}_\kappa = \frac{EI}{L_u^{(1)}} \begin{bmatrix} \mathbf{0} & \mathbf{I} & \mathbf{0} & -\mathbf{I} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -\mathbf{I} & \mathbf{0} & \mathbf{I} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \tag{A.9}$$

The first element consistent torsional stiffness matrix is:

$$\mathbf{K}_\tau = \tag{A.10}$$
$$\frac{GJ\tau^{(1)}}{L_u^{(1)}} \begin{bmatrix} \frac{1}{2}\left(\mathbf{Q}^{(0)}+\mathbf{Q}^{(1)}\right) & -\frac{(L_u^{(1)})^2}{24}\mathbf{Q}^{(1)} & -\frac{1}{2}\left(\mathbf{Q}^{(0)}+\mathbf{Q}^{(1)}\right) & \frac{(L_u^{(1)})^2}{24}\mathbf{Q}^{(0)} \\ \frac{(L_u^{(1)})^2}{24}\left(\mathbf{Q}^{(0)}-\mathbf{Q}^{(1)}\right) & \frac{(L_u^{(1)})^4}{120}\mathbf{Q}^{(1)} & -\frac{(L_u^{(1)})^2}{24}\left(\mathbf{Q}^{(0)}-\mathbf{Q}^{(1)}\right) & \frac{7(L_u^{(1)})^4}{720}\mathbf{Q}^{(0)} \\ -\frac{1}{2}\left(\mathbf{Q}^{(0)}+\mathbf{Q}^{(1)}\right) & \frac{(L_u^{(1)})^2}{24}\mathbf{Q}^{(1)} & \frac{1}{2}\left(\mathbf{Q}^{(0)}+\mathbf{Q}^{(1)}\right) & -\frac{(L_u^{(1)})^2}{24}\mathbf{Q}^{(0)} \\ \frac{(L_u^{(1)})^2}{24}\left(\mathbf{Q}^{(0)}-\mathbf{Q}^{(1)}\right) & \frac{7(L_u^{(1)})^4}{720}\mathbf{Q}^{(1)} & -\frac{(L_u^{(1)})^2}{24}\left(\mathbf{Q}^{(0)}-\mathbf{Q}^{(1)}\right) & \frac{(L_u^{(1)})^4}{120}\mathbf{Q}^{(0)} \end{bmatrix}$$

where $\tau^{(1)}$ is the first element torsion and the matrices $\mathbf{Q}^{(0)}$ and $\mathbf{Q}^{(1)}$ represent the skew-symmetric cross product matrix formed from the components of the node curvature values. For example:

$$\mathbf{Q}^{(0)} = \begin{bmatrix} 0 & -r_z''^{(0)} & r_y''^{(0)} \\ r_z''^{(0)} & 0 & -r_x''^{(0)} \\ -r_y''^{(0)} & r_x''^{(0)} & 0 \end{bmatrix} \tag{A.11}$$

The consistent form of the weight and buoyancy $\mathbf{W}^{(1)}$ forces, half of which are lumped to the adjacent nodes, are:

$$\mathbf{W}^{(1)} = L_u^{(1)} \frac{\pi d_c^2}{4} \left(\rho_c - \rho_w\right) \begin{bmatrix} \frac{1}{2}\mathbf{g} \\ \frac{\left(-L_u^{(1)}\right)^2}{24} \\ \frac{1}{2}\mathbf{g} \\ \frac{-\left(L_u^{(1)}\right)}{24} \end{bmatrix} \tag{A.12}$$

where $\mathbf{g}$ is the gravitational acceleration vector and ρ_w is the density of sea water. The consistent hydrodynamic $\mathbf{H}^{(1)}$ forces are:

$$\mathbf{H}^{(1)} = L_u^{(1)} \begin{bmatrix} \frac{1}{3}\mathbf{h}^{(0)} + \frac{1}{6}\mathbf{h}^{(1)} \\ -\frac{1}{45}\mathbf{h}^{(0)} - \frac{7}{360}\mathbf{h}^{(1)} \\ \frac{1}{6}\mathbf{h}^{(0)} + \frac{1}{3}\mathbf{h}^{(1)} \\ -\frac{7}{360}\mathbf{h}^{(0)} + -\frac{1}{45}\mathbf{h}^{(1)} \end{bmatrix} \tag{A.13}$$

where $\mathbf{h}^{(0)}$ is the distributed hydrodynamic load per unit length quantified at the node points with the assistance of a Morison-type approximation and the locally defined hydrodynamic frame.

After the lumped mass assumption, the reduced stiffness matrices have rows proportional to curvature acceleration removed. The first element lumped mass cable axial stiffness matrix is:

$$\mathbf{K}_{\epsilon} = \frac{EA\epsilon^{(1)}}{L_u^{(1)}} \begin{bmatrix} -\mathbf{I} & \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{I} & \mathbf{0} & -\mathbf{I} & \mathbf{0} \end{bmatrix} \tag{A.14}$$

The first element consistent bending stiffness matrix is:

$$\mathbf{K}_{\kappa} = \frac{EI}{L_u^{(1)}} \begin{bmatrix} \mathbf{0} & \mathbf{I} & \mathbf{0} & -\mathbf{I} \\ \mathbf{0} & -\mathbf{I} & \mathbf{0} & \mathbf{I} \end{bmatrix} \tag{A.15}$$

The first element lumped torsional stiffness matrix is:

$$\mathbf{K}_{\tau\ G} = \frac{GJ\tau^{(1)}}{L_u^{(1)}} \begin{bmatrix} \frac{1}{2}\left(\mathbf{Q}^{(0)} + \mathbf{Q}^{(1)}\right) & -\frac{(L_u^{(1)})^2}{24}\mathbf{Q}^{(1)} & -\frac{1}{2}\left(\mathbf{Q}^{(0)} + \mathbf{Q}^{(1)}\right) & \frac{(L_u^{(1)})^2}{24}\mathbf{Q}^{(0)} \\ -\frac{1}{2}\left(\mathbf{Q}^{(0)} + \mathbf{Q}^{(1)}\right) & \frac{(L_u^{(1)})^2}{24}\mathbf{Q}^{(1)} & \frac{1}{2}\left(\mathbf{Q}^{(0)} + \mathbf{Q}^{(1)}\right) & -\frac{(L_u^{(1)})^2}{24}\mathbf{Q}^{(0)} \end{bmatrix} \tag{A.16}$$

Finally, the lumped weight and buoyancy and hydrodynamic force vectors are:

$$\mathbf{W}_G^{(1)} = L_u^{(1)} \frac{\pi d_c^2}{4} \left(\rho_c - \rho_w\right) \begin{bmatrix} \frac{1}{2}\mathbf{g} \\ \frac{1}{2}\mathbf{g} \end{bmatrix} \tag{A.17}$$

$$\mathbf{H}_G^{(1)} = L_u^{(1)} \begin{bmatrix} \frac{1}{3}\mathbf{h}^{(0)} + \frac{1}{6}\mathbf{h}^{(1)} \\ \frac{1}{6}\mathbf{h}^{(0)} + \frac{1}{3}\mathbf{h}^{(1)} \end{bmatrix} \tag{A.18}$$

The stiffness matrices must be concatenated to form the global system stiffness matrix. For example, a system of $N = 3$ nodes produces the global stiffness matrices:

$$\mathbf{K}_{\epsilon\ G_{sys}} = EA \begin{bmatrix} -\frac{\epsilon^{(1)}}{L_u^{(1)}}\mathbf{I} & \mathbf{0} & \frac{\epsilon^{(1)}}{L_u^{(1)}}\mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \frac{\epsilon^{(1)}}{L_u^{(1)}}\mathbf{I} & \mathbf{0} & -\left(\frac{\epsilon^{(1)}}{L_u^{(1)}} + \frac{\epsilon^{(2)}}{L_u^{(2)}}\right)\mathbf{I} & \mathbf{0} & \frac{\epsilon^{(2)}}{L_u^{(2)}}\mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \frac{\epsilon^{(2)}}{L_u^{(2)}}\mathbf{I} & \mathbf{0} & -\frac{\epsilon^{(2)}}{L_u^{(2)}}\mathbf{I} & \mathbf{0} \end{bmatrix} \tag{A.19}$$

$$\mathbf{K}_{\kappa\ G_{sys}} = EI \begin{bmatrix} \mathbf{0} & \frac{1}{L_u^{(1)}}\mathbf{I} & \mathbf{0} & -\frac{1}{L_u^{(1)}}\mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -\frac{1}{L_u^{(1)}}\mathbf{I} & \mathbf{0} & \left(\frac{1}{L_u^{(1)}} + \frac{1}{L_u^{(2)}}\right)\mathbf{I} & \mathbf{0} & -\frac{1}{L_u^{(2)}}\mathbf{I} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & -\frac{1}{L_u^{(2)}}\mathbf{I} & \mathbf{0} & \frac{1}{L_u^{(2)}}\mathbf{I} \end{bmatrix} \tag{A.20}$$

where

$$\mathbf{K}_{\tau\ G_{sys}} = GJ \begin{bmatrix} \mathbf{K}_{\tau\ G_{sys}}^{11} & \mathbf{K}_{\tau\ G_{sys}}^{12} & \mathbf{K}_{\tau\ G_{sys}}^{13} & \mathbf{K}_{\tau\ G_{sys}}^{14} & \mathbf{0} & \mathbf{0} \\ \mathbf{K}_{\tau\ G_{sys}}^{21} & \mathbf{K}_{\tau\ G_{sys}}^{22} & \mathbf{K}_{\tau\ G_{sys}}^{23} & \mathbf{K}_{\tau\ G_{sys}}^{24} & \mathbf{K}_{\tau\ G_{sys}}^{25} & \mathbf{K}_{\tau\ G_{sys}}^{26} \\ \mathbf{0} & \mathbf{0} & \mathbf{K}_{\tau\ G_{sys}}^{33} & \mathbf{K}_{\tau\ G_{sys}}^{34} & \mathbf{K}_{\tau\ G_{sys}}^{35} & \mathbf{K}_{\tau\ G_{sys}}^{36} \end{bmatrix} \tag{A.21}$$

$$\mathbf{K}^{11}_{\tau\ G_{sys}} = \frac{\tau^{(1)}}{2L_u^{(1)}}\left(\mathbf{Q}^{(0)} + \mathbf{Q}^{(1)}\right) \tag{A.22}$$

$$\mathbf{K}^{12}_{\tau\ G_{sys}} = -\frac{\tau^{(1)}L_u^{(1)}}{24}\mathbf{Q}^{(1)} \tag{A.23}$$

$$\mathbf{K}^{13}_{\tau\ G_{sys}} = -\frac{\tau^{(1)}}{2L_u^{(1)}}\left(\mathbf{Q}^{(0)} + \mathbf{Q}^{(1)}\right) \tag{A.24}$$

$$\mathbf{K}^{14}_{\tau\ G_{sys}} = \frac{\tau^{(1)}L_u^{(1)}}{24}\mathbf{Q}^{(0)} \tag{A.25}$$

$$\mathbf{K}^{21}_{\tau\ G_{sys}} = -\frac{\tau^{(1)}}{2L_u^{(1)}}\left(\mathbf{Q}^{(0)} + \mathbf{Q}^{(1)}\right) \tag{A.26}$$

$$\mathbf{K}^{22}_{\tau\ G_{sys}} = \frac{\tau^{(1)}(L_u^{(1)})}{24}\mathbf{Q}^{(1)} \tag{A.27}$$

$$\mathbf{K}^{23}_{\tau\ G_{sys}} = \frac{1}{2}\left(\frac{\tau^{(1)}}{L_u^{(1)}}\mathbf{Q}^{(0)} + \left[\frac{\tau^{(1)}}{L_u^{(1)}} + \frac{\tau^{(2)}}{L_u^{(2)}}\right]\mathbf{Q}^{(1)} + \frac{\tau^{(2)}}{L_u^{(2)}}\mathbf{Q}^{(2)}\right) \tag{A.28}$$

$$\mathbf{K}^{24}_{\tau\ G_{sys}} = -\frac{\tau^{(1)}(L_u^{(1)})}{24}\mathbf{Q}^{(0)} - \frac{\tau^{(2)}L_u^{(2)}}{24}\mathbf{Q}^{(1)} \tag{A.29}$$

$$\mathbf{K}^{25}_{\tau\ G_{sys}} = -\frac{\tau^{(2)}}{2L_u^{(2)}}\left(\mathbf{Q}^{(1)} + \mathbf{Q}^{(2)}\right) \tag{A.30}$$

$$\mathbf{K}^{26}_{\tau\ G_{sys}} = \frac{\tau^{(2)}L_u^{(2)}}{24}\mathbf{Q}^{(1)} \tag{A.31}$$

$$\mathbf{K}^{33}_{\tau\ G_{sys}} = -\frac{\tau^{(2)}}{2L_u^{(2)}}\left(\mathbf{Q}^{(1)} + \mathbf{Q}^{(2)}\right) \tag{A.32}$$

$$\mathbf{K}^{34}_{\tau\ G_{sys}} = \frac{\tau^{(2)}L_u^{(2)}}{24}\mathbf{Q}^{(2)} \tag{A.33}$$

$$\mathbf{K}^{35}_{\tau\ G_{sys}} = \frac{\tau^{(2)}}{2L_u^{(2)}}\left(\mathbf{Q}^{(1)} + \mathbf{Q}^{(2)}\right) \tag{A.34}$$

$$\mathbf{K}^{36}_{\tau\ G_{sys}} = -\frac{\tau^{(2)}L_u^{(2)}}{24}\mathbf{Q}^{(1)} \tag{A.35}$$

$$\tag{A.36}$$

Finally, the lumped weight and buoyancy and hydrodynamic force vectors are:

$$\mathbf{W}^{(1)}_{G_{sys}} = \frac{\pi d_c^2}{4}\left(\rho_c - \rho_w\right)\begin{bmatrix} \frac{L_u^{(1)}}{2}\mathbf{g} \\ \frac{L_u^{(1)}+L_u^{(2)}}{2}\mathbf{g} \\ \frac{L_u^{(2)}}{2}\mathbf{g} \end{bmatrix} \tag{A.37}$$

$$\mathbf{H}^{(1)}_{G_{sys}} = \begin{bmatrix} L_u^{(1)}\left(\frac{1}{3}\mathbf{h}^{(0)} + \frac{1}{6}\mathbf{h}^{(1)}\right) \\ \frac{1}{6}L_u^{(1)}\mathbf{h}^{(0)} + \frac{2}{3}\mathbf{h}^{(1)}\left(L_u^{(1)} + L_u^{(2)}\right) + \frac{1}{6}L_u^{(2)}\mathbf{h}^{(2)} \\ L_u^{(2)}\left(\frac{1}{6}\mathbf{h}^{(1)} + \frac{1}{3}\mathbf{h}^{(2)}\right) \end{bmatrix} \tag{A.38}$$

This page intentionally left blank.

Annex B: Element Torsion Matrices

The tridiagonal system of equations that, when solved simultaneously, provide the node twist angle α that is used to calculate the mechanical torsion of a system of $N = 3$ nodes are:

$$GJ \begin{bmatrix} \frac{1}{L_u^{(1)}} & -\frac{1}{L_u^{(1)}} & 0 \\ -\frac{1}{L_u^{(1)}} & \frac{1}{L_u^{(1)}} + \frac{1}{L_u^{(2)}} & 0 \\ 0 & -\frac{1}{L_u^{(2)}} & \frac{1}{L_u^{(2)}} \end{bmatrix} \begin{bmatrix} \alpha^{(0)} \\ \alpha^{(1)} \\ \alpha^{(2)} \end{bmatrix} = GJ \begin{bmatrix} \gamma^{\left(\frac{1}{2}\right)} \\ \gamma^{\left(1+\frac{1}{2}\right)} - \gamma^{\left(\frac{1}{2}\right)} \\ -\gamma^{\left(2-\frac{1}{2}\right)} \end{bmatrix} \tag{B.1}$$

where $\gamma^{\left(\frac{1}{2}\right)}$ is the torsion angle of the Frenet frame at node 0, and $\gamma^{\left(1+\frac{1}{2}\right)} - \gamma^{\left(\frac{1}{2}\right)}$ is the torsion angle of the Frenet frame at node 1 relative to node 0.

This page intentionally left blank.

Annex C: Galerkin Method of Weighted Residuals

C.1 Galerkin Criterion

The Galerkin Method of Weighted Residuals (GMWR) is used here to derive the element equations for the finite element cable model used by the `SMS::Cable` as presented by the authors of [4, 5]. In effect, the method provides lumped equivalents of the continuously distributed hydrodynamic, inertia and internal forces at the element nodes. The method requires an equation of the form:

$$D(f(r(s,t),s)) = \cancelto{0}{G(s)} \tag{C.1}$$

where the function of interest, $f(r(s,t),s)$, is a continuous function over the domain and is acted on by D, a linear differential operator.

The function $r(s)$ can be approximated as $\tilde{r}(s)$, which is a linear combination of basis, or shape, functions $\phi_m(s)$ such that:

$$r(s,t) \approx \tilde{r}(s,t) = \sum_{m=1}^{n} \sigma_m(t)\phi_m(s) \tag{C.2}$$

where $\sigma_m(t)$ are unknown time dependent nodal variables. For the `SMS::Cable` class, these nodal variables are the absolute positions and curvatures of the cable at the nodes.

The difference between the approximation and the true solution, the residual, is described as:

$$R(s) = D(f(\tilde{r}(s,t),s)) \neq 0 \tag{C.3}$$

The method of weighted residuals seeks values for $\sigma_m(t) \quad (m = 1, 2, ..., n)$ that drive weighted averages of the residual to 0 over the modelled domain, such that:

$$\int_s R(s,t)W_m ds = 0, \qquad for \quad m = 1, 2, ..., n \tag{C.4}$$

where s is the domain and W_m are the weighting functions. There are an equal number of weighting functions to the number of nodal variables $\sigma_m(t)$ and so the series of equations in C.4 is deterministic. Galerkin's criterion follows the premise that the optimal weighting functions are the shape functions used in the approximation $\tilde{r}(s,t)$. That is:

$$W_m = \phi_m(s), \quad for \;\; m = 1, 2, ..., n \tag{C.5}$$

C.2 Galerkin Criterion and the Cable Model

The cable element model used in the `SMS::Cable` class, has equations of motion that are generally of the form:

$$\mathbf{f}(\mathbf{r}(s,t),s) = \mathbf{K}(\mathbf{r}(s,t)) + \mathbf{h}(\mathbf{r}(s,t),\dot{r}(s,t)) + \mathbf{w}(s) - \mathbf{I}(\mathbf{r}(s,t),\ddot{\mathbf{r}}(s,t)) = \mathbf{0} \tag{C.6}$$

where $\mathbf{K}$ represents the internal forces in the cable at a location s along the cable, $\mathbf{h}$ represents the hydrodynamics forces, $\mathbf{w}$ represents the differential weight and buoyancy forces at a location s along the cable, and $\mathbf{I}$ represents the inertia forces at a location s along the cable.

The true cable position vector $\mathbf{r}(s)$ is approximated using (see Equation 1)

$$\tilde{\mathbf{r}}(s) = \mathbf{r}^{(i-1)}\phi_{i,1} + \mathbf{r}''^{(i-1)}\phi_{i,2} + \mathbf{r}^{(i)}\phi_{i,3} + \mathbf{r}''^{(i)}\phi_{i,4} \tag{C.7}$$

where $(\cdot)_i$ refers to the i^{th} element, $(\cdot)^{(i)}$ refers to the i^{th} node, $\mathbf{r}$ is the absolute position of the cable centerline, and $\mathbf{r}''$ refers to the second derivative of the absolute position of the cable centerline derived with respect to s.

In applying the Galerkin method of weighted residuals, the residual is weighted, integrated over the domain, and forced to 0 to take the form:

$$\int_{s^{(i-1)}}^{s^{(i)}} \mathbf{R}(s,t)W_m ds = \int_{s^{(i-1)}}^{s^{(i)}} D(\mathbf{K}(\tilde{\mathbf{r}}(s,t)) + \mathbf{h}(\tilde{\mathbf{r}}(s,t),\dot{\tilde{r}}(s,t)) + \mathbf{w}(s) - \mathbf{I}(\tilde{\mathbf{r}}(s,t),\ddot{\tilde{r}}(s,t)))W_m ds = 0 \tag{C.8}$$

Equation C.8 is evaluated analytically to yield algebraic equations in terms of the nodal variables, $\mathbf{r}^{(i-1)}$, $\mathbf{r}''^{(i-1)}$, $\mathbf{r}^{(i)}$, $\mathbf{r}''^{(i)}$ and the corresponding velocities and accelerations.

These equations are coded within `SMS::Cable` and are solved explicitly for the values of nodal accelerations, using the nodal position and velocity initial conditions, at each timestep. The new found accelerations get integrated over time to provide the new velocities and position of the nodes of the `SMS::Cable`.

DOCUMENT CONTROL DATA

(Security markings for the title, abstract and indexing annotation must be entered when the document is Classified or Designated.)

1. ORIGINATOR (The name and address of the organization preparing the document. Organizations for whom the document was prepared, e.g. Centre sponsoring a contractor's report, or tasking agency, are entered in section 8.)

Dynamic Systems Analysis Limited
101-19 Dallas Road, Victoria, BC, Canada, V8V 5A6

2a. SECURITY MARKING (Overall security marking of the document, including supplemental markings if applicable.)

UNCLASSIFIED

2b. CONTROLLED GOODS

(NON-CONTROLLED GOODS)
DMC A
REVIEW: GCEC APRIL 2011

3. TITLE (The complete document title as indicated on the title page. Its classification should be indicated by the appropriate abbreviation (S, C or U) in parentheses after the title.)

Ship Mechanical Systems API – Final Report: Modeling Methodologies, API User Manual, API Validation

4. AUTHORS (Last name, followed by initials – ranks, titles, etc. not to be used.)

Roy, A.; Steinke, D.; Nicoll, R.

5. DATE OF PUBLICATION (Month and year of publication of document.)	6a. NO. OF PAGES (Total containing information. Include Annexes, Appendices, etc.)	6b. NO. OF REFS (Total cited in document.)
January 2014	270	43

7. DESCRIPTIVE NOTES (The category of the document, e.g. technical report, technical note or memorandum. If appropriate, enter the type of report, e.g. interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered.)

Contract Report

8. SPONSORING ACTIVITY (The name of the department project office or laboratory sponsoring the research and development – include address.)

Defence Research and Development Canada – Atlantic
PO Box 1012, Dartmouth NS B2Y 3Z7, Canada

9a. PROJECT OR GRANT NO. (If appropriate, the applicable research and development project or grant number under which the document was written. Please specify whether project or grant.)	9b. CONTRACT NO. (If appropriate, the applicable number under which the document was written.)
11ge02	W7707-115188/001/HAL

10a. ORIGINATOR'S DOCUMENT NUMBER (The official document number by which the document is identified by the originating activity. This number must be unique to this document.)	10b. OTHER DOCUMENT NO(s). (Any other numbers which may be assigned this document either by the originator or by the sponsor.)
DRDC Atlantic CR 2010-256	

11. DOCUMENT AVAILABILITY (Any limitations on further dissemination of the document, other than those imposed by security classification.)

(X) Unlimited distribution
( ) Defence departments and defence contractors; further distribution only as approved
( ) Defence departments and Canadian defence contractors; further distribution only as approved
( ) Government departments and agencies; further distribution only as approved
( ) Defence departments; further distribution only as approved
( ) Other (please specify):

12. DOCUMENT ANNOUNCEMENT (Any limitation to the bibliographic announcement of this document. This will normally correspond to the Document Availability (11). However, where further distribution (beyond the audience specified in (11)) is possible, a wider announcement audience may be selected.)

Unlimited

13. ABSTRACT (A brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C), or (U). It is not necessary to include here abstracts in both official languages unless the text is bilingual.)

This report describes development of a new software library for modelling ship mechanical systems. The library can model cables, winches, cables, and payloads. The library is also able to model collections of these components working together. When developing the library, computational fidelity was of primary importance and computational speed was of secondary importance. This initial version of the software library includes modelling of collisions and associated contact forces. The library is intended for modelling of naval platform systems, such as launch and recovery of a small craft from a ship.

14. KEYWORDS, DESCRIPTORS or IDENTIFIERS (Technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible keywords should be selected from a published thesaurus. e.g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus identified. If it is not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title.)

collision dynamics
cranes
launch and recovery
ship motions
simulation
time domain
towing
waves

This page intentionally left blank.

Defence R&D Canada

Canada's leader in defence
and National Security
Science and Technology

R & D pour la défense Canada

Chef de file au Canada en matière
de science et de technologie pour
la défense et la sécurité nationale

DEFENCE DÉFENSE

www.drdc-rddc.gc.ca