by CM LABS vortex

Theory Guide:
Vortex Software's Multibody Dynamics Engine

This document introduces the theory underlying Vortex Software's multibody dynamics engine. The goal is to provide you with information that will guide you toward good modeling decisions.

1 INTRODUCTION

In the last decades, interactive, physics-based simulation has become an established tool in the industry for the purposes of mission planning, operator training, and product design. Applications range from control familiarisation, increase in operator efficiency, and preparation for unforeseen circumstances, to virtual human-in-the-loop product design and development.

Only physically based modeling approaches can tackle the challenges involved in many related simulation scenarios, and accurately capture the complexities of interactions between machines and their environment.

Vortex Software's unique mix of accuracy, stability, and speed makes it an effective and reliable tool for the creation of high-fidelity, interactive, physics-based simulations.

The purpose of this document is to introduce you to the theory that underlies the Vortex multibody dynamics engine. A basic understanding of the methods used can help you make good modeling decisions, and choose simulation parameters that will lead to faster and more reliable simulations.

In Section 2, Vortex Software's multibody dynamics formulation is derived, starting with the introduction of rigid bodies and the Newton-Euler equations of motion. We introduce you to a modular constraint-based modeling that allows creation of complex machinery interacting with its environment. The characteristics of unilateral contacts with friction are formulated as a linear complementarity problem (LCP), and we'll discuss efficient solution techniques.

Sections 3, 4, and 5 focus on a series of Vortex Software modules that facilitate the creation of hoisting systems, flexible cables, vehicles, and earth-moving equipment interacting with deformable soil.

We'll conclude with Section 6, which provides details on the simulation of Vortex Software rigid bodies immersed in fluids.

Throughout this document we'll show how Vortex Software modules make use of the modeling framework presented in Section 2 to create a fully coupled physical environment.

It will become clear that through the use of rigid bodies and constraints, complex behaviour naturally emerges from the simulation, and leads to realistic physical interactions between all simulated entities.

2 Multibody Dynamics Simulation

All Vortex Software simulations involve a set of rigid bodies and constraints. The rigid bodies represent the moving components of a mechanism, such as the wheels in a vehicle or the segments in a robotic arm. Constraints (or joints) create restrictions in the relative motion of a subset of the rigid bodies. For example, a hinge will restrict the relative motion between two bodies to a rotation around a single axis, as in the hinge of a door.

Even the most complex mechanisms simulated with Vortex Software, including wheeled or tracked vehicles (with or without attached hoisting or earth-moving equipment) are modeled at the core as sets of constrained rigid bodies. Vortex Software modules, which will be presented in sections 3, 4, and 5, provide out-of-the-box capabilities for the simulation and modeling of cables, vehicles, and digging tools by pre-assembling rigid bodies with specialised constraints, all of which are processed by the Vortex Software simulation engine.

In Vortex Software, a simulation is advanced in a series of discrete time steps. A single time step is organised into several work stages which are processed in sequence. The workflow is shown in Figure **1**.

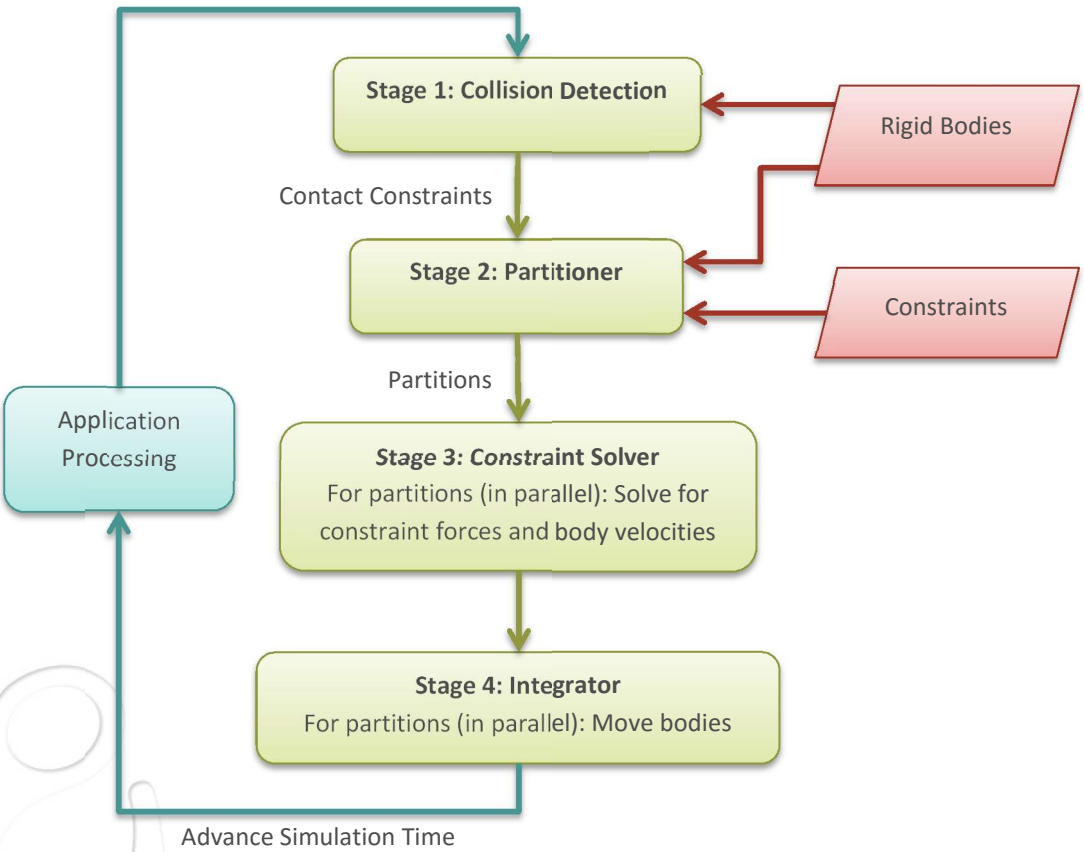

Figure 1. Simulation loop

Stage 1: Collision Detection

In the course of a simulation, rigid bodies can move and consequently collide. In order to model the physical interactions between colliding rigid bodies, contact constraints are automatically generated and added to the simulation. This is done during Stage 1 of every simulation step, the collision detection stage.

For the purpose of detecting collisions, a rigid body comes with a set of attached collision geometries [1], such as primitives or triangle meshes, which define the body's shape. Overlaps of collision geometries are detected, as well as spatial intersection information, such as exact touching locations and penetration depths or overlap volumes. An example is shown in Figure **2**. This information is then used to set up contact constraints that model the physical nature of the rigid body collision via non-penetration and friction conditions detailed in the *Modeling Contacts and Friction* section of this guide.

A naive approach for identifying all intersections between a set of n collision geometries is to perform intersection tests between any geometry and the remaining $n-1$ geometries. The computational complexity of this approach is $O(n^2)$, which makes it unsuitable even for moderate numbers of collision geometries.

In Vortex Software's multibody dynamics engine, a more efficient approach greatly reduces the computational complexity. The collision detection algorithm proceeds by performing a series of overlap tests, beginning with fast, efficient, approximate overlap tests on bounding volumes of the collision geometries, followed by progressively more accurate and less conservative tests. As a result of this process, a small set of potentially overlapping pairs of collision geometries is identified. Exact intersection tests are performed only for these pairs, which significantly reduces the computational effort compared to the brute force approach outlined above. For more information on this approach, see [2].

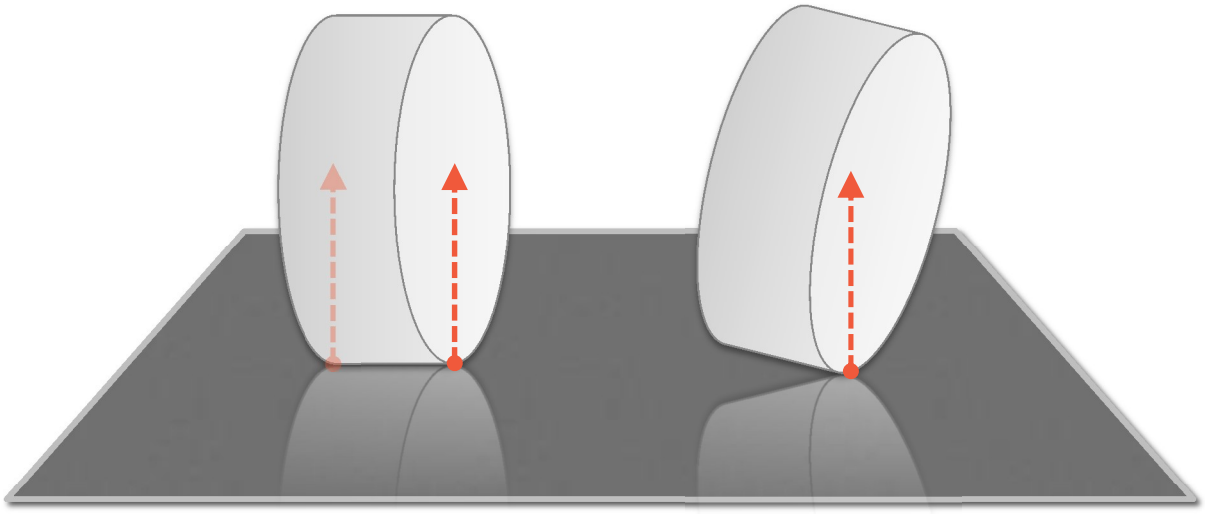

Figure 2. Collision: Simple example of a primitive cylinder (which could represent a wheel of a vehicle), colliding with a planar surface. Depending on the configuration, one or two contact points are generated in the collision detection, with contact normals as indicated.

Stage 2: Partitioner

All constraints (user-created as well as auto-generated contact constraints) are ultimately processed by the constraint solver (Stage 3 in Figure **1**), which computes the forces and torques applied by the constraints on their respective rigid bodies.

In preparation for this stage, the partitioner (Stage 2) organises the set of rigid bodies into multiple subsets — so-called partitions. Every partition contains a subset of the rigid bodies together with their related joints. Partitions are automatically computed. Groups of rigid bodies that are entirely isolated from the rest (meaning that they affect only each other via constraints) are placed into separate partitions.

As an example, consider two cars driving at different locations on a street. The rigid bodies of each car are constrained relative to each other via joints (e.g., the attachment of the wheels on the chassis), or are colliding with the street (e.g., the wheels). But the cars do not collide with each other and are also in no other way linked together. Therefore the subset of rigid bodies of each car will be organised into separate, isolated partitions.

A single constraint can connect rigid bodies in a single partition, or across different partitions. In the latter case, the respective partitions are coupled. Partitions that are not coupled (meaning that there is no constraint that connects a rigid body in those partitions to a rigid body in any other partition, as is the

case in the car example) represent independent simulation problems, and will be processed in parallel in the next stage.

Partitions that are coupled (meaning that there exists a constraint connecting them to another partition), will affect each other in simulation. In the car example, this would occur if the vehicles collided, in which case a contact constraint would be created. Based on the configuration of the partitioner and the desired simulation quality and speed, the partitions of the colliding cars could now be combined into a single, larger partition. They could alternatively be kept separate but considered as coupled partitions. In the latter case, the force exchange between the two cars would be handled in the following stage.

Stages 3 and 4: Constraint Solver and Integrator

The constraint solver (Stage 3) receives the partitions, and processes them to compute the forces applied by the constraints involved. Isolated, uncoupled partitions represent independent and data-parallel problems, which are processed in parallel. This greatly reduces the time consumption in the constraint solver.

The forces exchanged at the interaction boundaries of coupled partitions are computed in an iterative way. In this approach, coupled partitions are initially solved isolated and in parallel, and interaction forces are computed. The interaction forces are applied to the rigid bodies at the interaction boundaries followed by another solve. This procedure is repeated iteratively a given number of times. Once the constraint forces are computed, the rigid bodies can be moved to their new respective locations at the end of the current simulation step. This is done during Stage 4 for each partition in parallel.

It should be noted that multiple hooks into the presented simulation loop are provided. Among others, a client application can receive collision notifications during the collision detection stage, or obtain access to the simulation data passed to the constraint solver [3]. As such, a simulation created using Vortex Software can be highly customised and tailored to the client's needs, for example with specialised collision handling or custom constraint implementations [3]. All Vortex Software modules (i.e., Earthwork Systems, Vehicle Systems, and Cable Systems modules) make heavy use of this technology.

In the following sections, Stages 3 and 4 are investigated in more detail. The constraint solver and its theoretical basis are presented, starting with an exploration of the motion of rigid bodies and the introduction of constraints. Later, the developed constraint framework is applied to the modeling of unilateral contacts with friction.

2.1 Dynamics of Rigid Bodies

The Newton-Euler equations from classical mechanics describe the dynamics of a rigid body, and were derived by Euler as an extension of Newton's second law, by viewing a rigid body as a set of infinitesimally small point masses [4]. Accordingly, the net force and torque, $\mathbf{F}$ and $\boldsymbol{\tau}$, applied on a single rigid body moving with linear and angular velocities $\mathbf{v}$ and $\boldsymbol{\omega}$ respectively and undergoing linear and angular accelerations $\dot{\mathbf{v}}$ and $\dot{\boldsymbol{\omega}}$, is defined by the following ordinary differential equations:

$$\begin{aligned} \mathbf{F} &= m\dot{\mathbf{v}} \\ \boldsymbol{\tau} &= \mathbf{I}\,\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times \mathbf{I}\boldsymbol{\omega} \end{aligned} \tag{1}$$

Here, m defines the scalar mass of the body and $\mathbf{I}$ its inertia tensor. All quantities are specified with respect to the inertial frame of reference, which is a fixed and global coordinate frame.

By introducing the 6×1 generalised velocity vector $\mathbf{u}_i^T = [\mathbf{v}_i^T \;\; \boldsymbol{\omega}_i^T]$ for rigid body i, and collecting the generalised velocities for a set of n rigid bodies in the $6n \times 1$ vector $\mathbf{u}$, the above equations can be assembled in matrix form as

$$\mathbf{M}\dot{\mathbf{u}} = \mathbf{g} \tag{2}$$

where $\mathbf{M}$ is the $6n \times 6n$ block-diagonal mass matrix with 6×6 block elements $\mathbf{M}_i$ containing the mass properties of the ith rigid body as shown below:

$$\mathbf{M} = \begin{bmatrix} \mathbf{M}_1 & \mathbf{0} & \dots & 0 \\ 0 & \ddots & & \vdots \\ \vdots & & & 0 \\ 0 & \dots & 0 & \mathbf{M}_n \end{bmatrix}, \qquad \mathbf{M}_i = \begin{bmatrix} m_i\mathbf{1}_{3\times 3} & 0 \\ 0 & \mathbf{I}_i \end{bmatrix} \tag{3}$$

The $6n \times 1$ vector $\mathbf{g}$ is the vector of generalised forces and consists of the net load and the gyroscopic moment for all rigid bodies:

$$\mathbf{g}^T = [\dots \quad \mathbf{F}_i{}^T, \quad (\boldsymbol{\tau}_i - \boldsymbol{\omega}_i \times \mathbf{I}_i\boldsymbol{\omega}_i)^T \quad \dots] \tag{4}$$

In the inertial frame of reference, the quantities $\mathbf{M}$ and $\mathbf{g}$ are time-varying and depend on the current configuration of the rigid bodies, and, in the case of $\mathbf{g}$, also on the current angular velocities, as can be seen in equation (4).

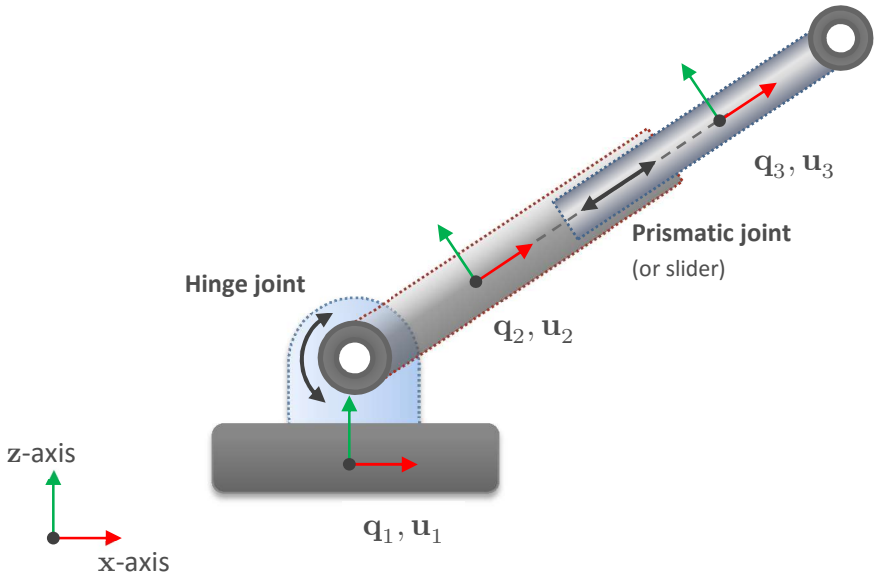

Figure 3. A system of three articulated bodies connected by a hinge and a prismatic joint.

2.2 Constraints

We introduce constraints to equation (2) in order to model interactions between individual rigid bodies, such as mechanical joint articulations and contacts. An example of an articulated assembly of rigid bodies is shown in Figure **3**. The rigid bodies are connected by joints that restrict their ability to move relative to each other. The figure depicts a simplified hydraulic cylinder which is modeled by a prismatic joint (or slider) that permits one linear degree of freedom (DOF). The cylinder is attached to the ground using a hinge joint with one angular DOF.

Joints, such as the prismatic or hinge joint, are modeled using constraints. The number of constraints required to model a particular joint depends on the restrictions the joint applies on the positions and orientations of the involved rigid bodies relative to each other. As an example, consider a two-body joint. Relative to each other, two bodies have six degrees of freedom: Three linear and three angular. A hinge or prismatic joint permits only one degree of freedom, as shown in Figure **3**, which means that five constraints are required to represent these joint types.

Constraints are equalities or inequalities of scalar functions ϕ restricting the motion of two or more rigid bodies relative to each other, and have the following form:

$$\begin{aligned}\phi(\mathbf{q}_{j_1}, \mathbf{u}_{j_1}, \ldots, \mathbf{q}_{j_k}, \mathbf{u}_{j_k}, t) &= 0 \\ \phi(\mathbf{q}_{j_1}, \mathbf{u}_{j_1}, \ldots, \mathbf{q}_{j_k}, \mathbf{u}_{j_k}, t) &\geq 0\end{aligned} \tag{5}$$

where $j_i \in \{1, \ldots, n\}$ denote the indices of the k constrained bodies and $\mathbf{q}_{j_i}$ their respective generalised position.

The generalised position $\mathbf{q}_i$ provides the position and orientation of the ith rigid body in the system. It is related to the body's generalised velocity $\mathbf{u}_i$ through the expression

$$\dot{\mathbf{q}}_i = \mathbf{H}(\mathbf{q}_i)\mathbf{u}_i \tag{6}$$

The matrix $\mathbf{H}(\mathbf{q}_i)$ is called the kinematic map and has the following form:

$$\mathbf{H}(\mathbf{q}_i) = \begin{bmatrix}\mathbf{1}_{3\times 3} & \mathbf{0} \\ \mathbf{0} & \mathbf{G}(\mathbf{q}_i)\end{bmatrix} \tag{7}$$

The term $\mathbf{G}(\mathbf{q}_i)$ is dependent on the generalised positions and the choice of coordinates. A common way to represent the coordinates of a rigid body is to use a 3×1 position vector $\mathbf{x}_i$ together with a quaternion Q_i, represented as 4×1 vector, defining the body's orientation. This would make the generalised position $\mathbf{q}_i^T = [\mathbf{x}_i^T \;\; Q_i^T]$ a 7×1 vector and the term $\mathbf{G}(\mathbf{q}_i)$ a 4×3 matrix.

It should be noted that equality and inequality constraints are also commonly referred to as bilateral and unilateral constraints, respectively. As indicated above, a constraint can be dependent on time t, which is for example true for motorised constraints. Combining the Newton-Euler equations from equation (2) with a set of constraints yields a system of differential algebraic equations.

Constraints can be classified into holonomic and nonholonomic constraints. The latter class is comprised of all inequality constraints together with those equality constraints which are not integrable, meaning that derivatives of the generalised positions cannot be eliminated from the equation. Constraints can also contain higher-order derivatives of the generalised positions, e.g., acceleration terms. But since Vortex Software uses a velocity-based and not an acceleration-based formulation, they are not considered here.

An example of an inequality constraint is a unilateral contact. Contacts between rigid bodies are computed in the collision detection phase at the beginning of every time step, and are included in the simulation as contact constraints. Contact constraints will be discussed in more detail in the *Modeling*

Contacts and Friction section of this guide. The collision detection is responsible for detecting intersections between rigid bodies and producing a set of contact points representing the nature of the overlap between the bodies' geometries. The number of calculated contacts can range from one, in the simple case of two colliding spheres, up to any number, e.g., for intersecting meshes, based on the requested contact count and other contact discretisation settings [3].

During simulation, all constraints are enforced by the constraint solver. For example, under the assumption that at the current time the constraint $\phi = 0$ is satisfied, we know that the constraint will not be violated in the future if the time-derivative $\dot{\phi} = 0$ is maintained. Consequently, for a holonomic constraint of the form $\phi(\mathbf{q}) = 0$, where $\mathbf{q}^T = [\mathbf{q}_{j_1}^T, \cdots, \mathbf{q}_{j_k}^T]$, a kinematic constraint condition can be derived via differentiation with respect to time:

$$\frac{d}{dt}\phi(\mathbf{q}) = \frac{\partial \phi}{\partial \mathbf{q}}\frac{d\mathbf{q}}{dt} = \underbrace{\frac{\partial \phi}{\partial \mathbf{q}}\mathbf{H}(\mathbf{q})}_{\mathbf{J}_\phi}\mathbf{u} = \mathbf{J}_\phi \mathbf{u} = 0 \tag{8}$$

The row-vector $\mathbf{J}_\phi$ denotes the constraint Jacobian and is related to the spatial derivative of the generalised position constraint $\phi(\mathbf{q})$ as shown above. The term $\mathbf{H}(\mathbf{q})$ is a block-diagonal matrix with block-elements $\mathbf{H}(\mathbf{q}_{i_j})$ defined in equation (7), one block per constrained rigid body. From D'Alembert's principle it follows that the generalised impulse, $\mathbf{P}_\phi$, applied by a constraint ϕ on the constrained bodies, corresponds [5] to

$$\mathbf{P}_\phi = \mathbf{J}_\phi{}^T \lambda_\phi \tag{9}$$

where the scalar λ_ϕ denotes a Lagrange multiplier. In this context, the Lagrange multiplier represents the impulse in constraint space.

The following section will demonstrate how to use equation (9) to incorporate constraint impulses into the equations of motion, and how the kinematic constraint condition given in equation (8) is used to maintain constraints in the Vortex Software time stepper.

2.3 Discretisation of the Equations of Motion

In this section, we will discretise the equations of motion from equation (2) and derive a time-stepping scheme which approximates the evolution of the generalised velocities and positions of a system of constrained rigid bodies. We choose a discrete time step Δt and approximate $\dot{\mathbf{u}} = d\mathbf{u}/dt$ using backward Euler as $\dot{\mathbf{u}} \approx (\mathbf{u}_+ - \mathbf{u}_-)/\Delta t,$ where $-$ denotes the beginning of the current time step and $+$

denotes its end. This way a discretised version of equation (2) can be derived leading to the following expression:

$$\mathbf{M}\frac{(\mathbf{u}_+ - \mathbf{u}_-)}{\Delta t} = \mathbf{g} \quad (10)$$
$$\Leftrightarrow \qquad \mathbf{M}\mathbf{u}_+ = \Delta t\mathbf{g} + \mathbf{M}\mathbf{u}_-$$

The kinematic mapping from equation (6) can be discretised in an analogous way:

$$\mathbf{q}_+ = \mathbf{q}_- + \Delta t\mathbf{H}(\mathbf{q}_-)\mathbf{u}_+ \quad (11)$$

The term $\mathbf{H}(\mathbf{q}_-)$ is the generalised kinematic map, evaluated with respect to the generalised positions, $\mathbf{q}_-$, at the beginning of the current time step. It is defined analogously to $\mathbf{H}(\mathbf{q})$ from equation (7). By choosing $\mathbf{u}_+$ in the equation above, the generalised position update becomes implicit with respect to the generalised velocity, which has been shown to produce more stable results than a fully explicit update.

Finally we add the constraint impulses from equation (9) to the discrete-time Newton-Euler equations in equation (10) and incorporate the kinematic constraint condition from equation (8) derived in the previous section. In matrix form, this yields the following Karush-Kuhn-Tucker (KKT) system:

$$\begin{bmatrix}\mathbf{M} & -\mathbf{J}^T \\ \mathbf{J} & \mathbf{0}\end{bmatrix}\begin{bmatrix}\mathbf{u}_+ \\ \boldsymbol{\lambda}\end{bmatrix} = \begin{bmatrix}\Delta t\mathbf{g} + \mathbf{M}\mathbf{u}_- \\ \mathbf{v}_0\end{bmatrix} \quad (12)$$

The term **J** assembles all individual constraint Jacobian vectors $\mathbf{J}_\phi$ in a single $k \times 6n$ matrix where k is the total number of constraints and n the total number of rigid bodies.

For equation (12) to represent a linear system, all equations must be linear with respect to the sought generalised positions and velocities, $\mathbf{q}_+$ and $\mathbf{u}_+$, from the next time step [6]. This is the case if the time-varying terms $\mathbf{M}$, $\mathbf{J}$ and $\mathbf{g}$ are all evaluated at the beginning of the current time step, i.e., with respect to $\mathbf{q}_+$ and $\mathbf{u}_-$, which we will assume here.

In the above system we introduced a target velocity term $v_{0,\phi}$ for every constraint in the kinematic constraint condition, and assembled in the $k \times 1$ vector $\mathbf{v}_0$. Essentially, the target velocity allows the inducing of relative motion between two or more bodies. Among other things, this enables the modeling of time-dependent constraints such as motors. It can also be used to simulate other, more complex

kinematic relations between multiple bodies, such as differential gears, which comes in handy when modeling, e.g., vehicle transmissions (see the *Vehicle Systems* section of this guide).

As shown in equation (11), the generalised positions at the end of the current time step, $\mathbf{q}_+$, are computed implicitly based on the updated velocities $\mathbf{u}_+$. The generalised velocities $\mathbf{u}_+$ are computed explicitly if the quantities $\mathbf{M}$, $\mathbf{J}$ and $\mathbf{g}$ are evaluated at the beginning of the current time step, and implicitly if they are evaluated at the end of the current time step. In Vortex Software's multibody dynamics engine the latter approach is taken, which makes the presented time-stepping scheme semi-implicit. Moreover, it can be shown that the integrator is also symplectic.

2.4 Stabilisation and Relaxation

As shown in a previous section, in the Vortex Software integrator, constraints are enforced at the velocity level. Consequently, in a discretised system, constraint drift can arise at the position level. In order to prevent this drift, we apply constraint stabilisation based on the method from Baumgarte [7]. Accordingly, the constraint condition in equation (12) is modified and, for a single, holonomic and time-independent constraint ϕ, takes the form

$$\mathbf{J}_\phi \mathbf{u}_+ + \varepsilon_\phi \lambda_\phi = v_{0,\phi} - \gamma_\phi \frac{\phi(\mathbf{q}_-)}{\Delta t} \tag{13}$$

In the expression above, the term $v_{0,\phi}$ denotes the scalar target velocity for the constraint ϕ. The scalars ε_ϕ and γ_ϕ are the stabilisation parameters which are used to mix the constraint impulses λ_ϕ into the constraint function and account for the current scalar constraint violation $\phi(\mathbf{q}_-)$, respectively. As can be observed in the equation above, the stabilisation parameters offset the kinematic constraint condition which allows us to induce or counteract certain effects, such as positional drift. For more details, see [5].

The positional constraint correction term $\phi(\mathbf{q}_-)$ in equation (13) emerges from the Taylor series expansion of $\phi(\mathbf{q})$ around $\mathbf{q}_-$. After truncating any higher-order terms, this yields the approximation

$$\begin{aligned} \phi(\mathbf{q}_+) &\approx \phi(\mathbf{q}_-) + \frac{\partial \phi}{\partial \mathbf{q}} \Delta \mathbf{q} \\ &= \phi(\mathbf{q}_-) + \mathbf{J}_\phi \mathbf{u}_+ \Delta t \end{aligned} \tag{14}$$

where $\Delta \mathbf{q} = \Delta t \, \mathbf{H}(\mathbf{q}_-)\mathbf{u}_+$ according to (11).

Replacing the left-hand side of the constraint $\phi(\mathbf{q}_+) = 0$ by the approximation above and considering equations (8) and (11), we obtain the expression

$$\begin{aligned} \phi(\mathbf{q}_{-}) + \mathbf{J}_{\phi}\mathbf{u}_{+}\Delta t = 0 \\ \Leftrightarrow \ \mathbf{J}_{\phi}\mathbf{u}_{+} = -\frac{\phi(\mathbf{q}_{-})}{\Delta t} \end{aligned} \tag{15}$$

The equation above is identical to equation (13), except for the stabilisation terms and the offset velocity.

Apart from preventing constraint drift, the presented procedure has two other advantages. First, with $\varepsilon_{\phi} > 0$, the system matrix can be made positive definite, which leads to numerical stability and solvability, as we will see later. Second, by choosing appropriate stabilisation parameters, constraints can be relaxed, leading to softer constraint behaviour. As such, an important feature of the relaxation method applied in Vortex is that every positional constraint can obtain visco-elastic characteristics, and can be made equivalent to a spring-damper. This is achieved through a special mapping of the stiffness and damping coefficients in the Kelvin-Voigt model to the generic Baumgarte stabilisation parameters.

This is a significant benefit, as it is particularly hard to find good Baumgarte parameters for a given constraint [8], and one often has to resort to trial and error, which makes the procedure tedious and error-prone. Choosing stiffness and damping coefficients for a spring-damper, on the other hand, is an intuitive process because their physical effect is well understood, and the visco-elastic nature of the Kelvin-Voigt model is accurately reproduced in the Vortex Software multibody dynamics engine.

This important ability of constraints in Vortex Software allows the simulation of complex behaviours, such as the elastic deformations of flexible cables (see the *Mechanical Cable Properties* section of this document), or elasto-plastic soil deformations in a Vortex Earthwork Systems module simulation. It is also used to model visco-elastic collision response between rigid bodies. As a matter of fact, this ability [3] allows the specification of stiffness and damping coefficients for each pair of colliding surface types, such as steel and rubber, which enables the user to model collision responses ranging from very soft to very hard. In the case of pure velocity constraints, such as motors, the relaxation parameters are used to introduce loss, which allows us to control the amount of energy dissipation in the constraint.

Finally, the constraint stabilisation parameters are included in the system from equation (12), and a modification to the mass matrix $\mathbf{M}$ is performed that accounts for the evolution of the gyroscopic moment over the discrete time step, yielding the modified mass matrix $\widetilde{\mathbf{M}}$. This further improves the stability of the presented time stepper. Details can be found in [9]. Putting everything together, we obtain the final system of constrained dynamical equations

$$\begin{bmatrix} \widetilde{\mathbf{M}} & -\mathbf{J}^{T} \\ \mathbf{J} & \varepsilon \end{bmatrix} \begin{bmatrix} \mathbf{u}_{+} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \Delta t\mathbf{g} + \mathbf{M}\mathbf{u}_{-} \\ \mathbf{v}_{0} - \boldsymbol{\gamma}\dfrac{\boldsymbol{\phi}(\mathbf{q}_{-})}{\Delta t} \end{bmatrix} \tag{16}$$

with diagonal matrices ε and γ defined as $diag(\ldots \varepsilon_\phi \ldots)$and $diag(\ldots \gamma_\phi \ldots)$, respectively. The vector $\phi(\mathbf{q}_-)$ assembles the scalar constraint violations at the beginning of the next time step of all constraints.

As mentioned earlier, if the diagonal matrix ε is strictly positive, the system matrix in the above equation becomes positive definite and is thus invertible. This ensures that the system of linear equations is solvable using efficient numerical methods.

2.5 Modeling Contacts and Friction

Solving the system in equation (16) in order to find the constraint impulses $\boldsymbol{\lambda}$ and the future velocities $\mathbf{u}_+$ is not sufficient in certain situations. If the simulation contains inequality constraints, arising for example from contacts or friction, another approach must be taken. One such method is outlined in this section.

2.5.1 Contacts

For the purposes of modeling contacts, we define a gap function $g(\mathbf{q})$ which provides the signed distance between the surfaces of two rigid bodies at some contact point. The value of the gap function is zero when the bodies touch, negative when they overlap, and positive when they are separate, indicating a gap.

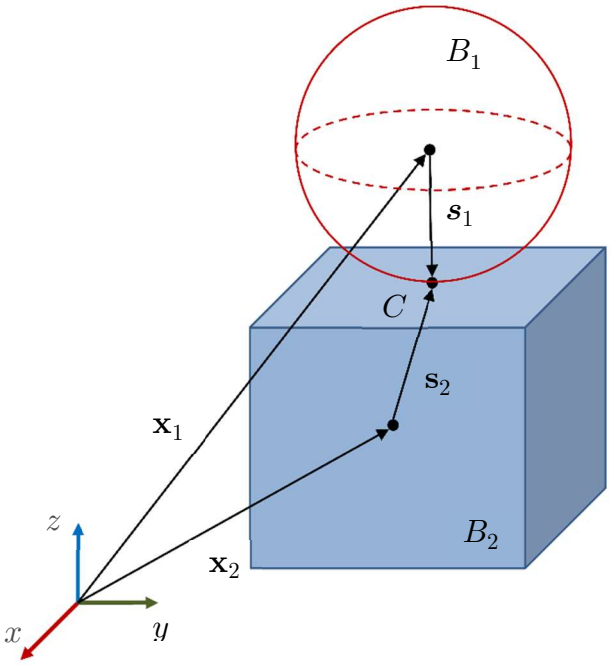

Figure 4. A single contact C between a sphere B_1 and a box primitive B_2.

Knowing that rigid bodies should not interpenetrate, a perfectly rigid contact can be defined as the scalar inequality constraint, or unilateral constraint

$$\phi_u(\mathbf{q}) = g(\mathbf{q}) \geq 0 \tag{17}$$

Given a contact situation as shown in Figure **4**, one possible gap function for the contact point C is

$$g(\mathbf{q}) = (\mathbf{x}_1 + \mathbf{s}_1 - \mathbf{x}_2 - \mathbf{s}_2) \cdot \mathbf{n} \tag{18}$$

where $\mathbf{n}$ denotes the surface normal at the contact point.

The time derivative of ϕ_u will be required in order to include the contact constraint from equation (17) into our simulation framework. It is given by

$$\begin{aligned} &\frac{d}{dt}\phi_u(\mathbf{q}) = \frac{d}{dt}\big((\mathbf{x}_1 + \mathbf{s}_1 - \mathbf{x}_2 - \mathbf{s}_2) \cdot \mathbf{n}\big) \\ &= \frac{d}{dt}(\mathbf{x}_1 + \mathbf{s}_1 - \mathbf{x}_2 - \mathbf{s}_2) \cdot \mathbf{n} + (\mathbf{x}_1 + \mathbf{s}_1 - \mathbf{x}_2 - \mathbf{s}_2) \\ &\cdot \frac{d}{dt}\mathbf{n} \\ &\approx (\mathbf{v}_1 + \boldsymbol{\omega}_1 \times \mathbf{s}_1 - \mathbf{v}_2 - \boldsymbol{\omega}_2 \times \mathbf{s}_2) \cdot \mathbf{n} \\ &= \underbrace{[\mathbf{n}^T \quad (\mathbf{s}_1 \times \mathbf{n})^T \quad -\mathbf{n}^T \quad (-\mathbf{s}_2 \times \mathbf{n})^T]}_{\mathbf{J}_{\phi_u}} \begin{bmatrix} \mathbf{v}_1 \\ \boldsymbol{\omega}_1 \\ \mathbf{v}_2 \\ \boldsymbol{\omega}_2 \end{bmatrix} \end{aligned} \tag{19}$$

In the expression above, the approximation arises from the assumption that the penetration depth is usually very small, in which case the contribution of the time derivative of the contact normal, $d\mathbf{n}/dt$, becomes insignificant.

It can be observed that a contact should induce an impulse only when the respective bodies touch, that is, when $\phi_u(\mathbf{q}) = 0$. On the other hand, when the bodies are separate, meaning that $\phi_u(\mathbf{q}) > 0$, the contact impulse will be zero. In mathematical terms, this means that the contact impulse, λ_{ϕ_u}, and the signed distance at the contact point, $\phi_u(\mathbf{q})$, are complementary variables. Another observation is that a rigid contact should not "stick," expressed as the requirement that the contact impulse must be exclusively positive or zero, that is, $\lambda_{\phi_u} \geq 0$.

Putting both observations together, the following complementarity condition for a unilateral contact can be derived:

$$0 \leq \phi_u(\mathbf{q}) \perp \lambda_{\phi_u} \geq 0 \tag{20}$$

If $\phi_u(\mathbf{q})$ is replaced by the approximation from equation (14), the condition above can be combined with the velocity integration scheme from equation (16). This is achieved by casting the dynamics equations and constraints into a mixed linear complementarity problem (MLCP).

In the general sense, an MLCP is defined as follows.

Definition 1. Mixed linear complementarity problem (MLCP): Given a known constant matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ *and constant vectors* $b, l, u \in \mathbb{R}^{n \times 1}$*, an MLCP denoted by* $MLCP(\mathbf{A}, \mathbf{b}, \mathbf{l}, \mathbf{u}, \mathbf{w})$ *consists of finding a solution vector* $\mathbf{x} \in \mathbb{R}^{n \times 1}$*, such that*

$$\begin{cases} \mathbf{A}\mathbf{x} + \mathbf{b} = \mathbf{w} = \mathbf{w}_+ - \mathbf{w}_- \\ \mathbf{0} \leq \mathbf{w}_+ \perp \mathbf{x} - \mathbf{l} \geq \mathbf{0} \\ \mathbf{0} \leq \mathbf{w}_- \perp \mathbf{u} - \mathbf{x} \geq \mathbf{0} \end{cases} \tag{21}$$

Here, $\mathbf{w}_+$ *and* $\mathbf{w}_-$ *denote the vectors which contain the positive and negative components of* $\mathbf{w}$*, respectively. The vector* $\mathbf{w}$ *is a slack variable used to turn inequalities in the system into equalities and is discarded once the solution is found. The vectors* $\mathbf{l}$ *and* $\mathbf{u}$ *respectively define component-wise lower and upper bounds on the solution vector* $\mathbf{x}$ *(including* $\pm\infty$*). Note that the complementarity operator* $\perp$ *is understood component-wise. In other words, if* $\langle \mathbf{x}, \mathbf{y} \rangle$ *denotes the inner product of vectors* $\mathbf{x}$ *and* $\mathbf{y}$*,* the expression $\mathbf{x} \perp \mathbf{y}$ *is equivalent to* $\langle \mathbf{x}, \mathbf{y} \rangle = 0$.

Given the above definition, a system of dynamics equations involving a set B of bilateral constraints of the form $\phi_b(\mathbf{q}) = 0$ with Jacobian matrix $\mathbf{J}_B$, and a set U of unilateral contact constraints $\phi_u(\mathbf{q}) \geq 0$ with Jacobian matrix $\mathbf{J}_U$ and gap function vector $\boldsymbol{g}_U \in \mathbb{R}^{|U| \times 1}$ can be formulated as $MLCP(\mathbf{A}, \mathbf{b}, \mathbf{l}, \mathbf{u}, \mathbf{w})$ with

$$\mathbf{A} = \begin{bmatrix} \widetilde{\mathbf{M}} & -\mathbf{J}_B{}^T & -\mathbf{J}_U{}^T \\ \mathbf{J}_B & \boldsymbol{\varepsilon}_B & \mathbf{0} \\ \mathbf{J}_U & \mathbf{0} & \boldsymbol{\varepsilon}_U \end{bmatrix}, \mathbf{x} = \begin{bmatrix} \mathbf{u}_+ \\ \boldsymbol{\lambda}_B \\ \boldsymbol{\lambda}_U \end{bmatrix} \tag{22}$$

$$\mathbf{b} = -\begin{bmatrix} \Delta t \mathbf{g} + \mathbf{M}\mathbf{u}_- \\ \mathbf{v}_0 - \boldsymbol{\gamma}_B \dfrac{\boldsymbol{\phi}_B(\mathbf{q}_-)}{\Delta t} \\ -\boldsymbol{\gamma}_U \dfrac{\boldsymbol{\phi}_U(\mathbf{q}_-)}{\Delta t} \end{bmatrix}, \mathbf{w} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ \boldsymbol{g}_U \end{bmatrix}$$

$$\mathbf{l} = \begin{bmatrix} -\infty \\ \vdots \\ -\infty \\ \mathbf{0}_{|U|} \end{bmatrix}, \mathbf{u} = \begin{bmatrix} +\infty \\ \vdots \\ +\infty \end{bmatrix}$$

where $\mathbf{0}_k$ denotes the zero vector $[0,\ldots,0]^T \in \mathbb{R}^{k\times 1}$. The vectors $\mathbf{w}$, $\mathbf{l}$, and $\mathbf{u}$ are carefully chosen such that only the unilateral contact constraints are modeled as inequalities with complementarity conditions, whereas the bilateral constraints remain equalities and do not appear in any complementarity condition.

It should be noted that the lower limits on the contact impulses $\boldsymbol{\lambda}_U \in \mathbb{R}^{|U|\times 1}$are set to zero in the vector $\mathbf{l}$, while the upper limits are unbounded. This ensures that contacts can only be compressive and do not "stick." Consequently, the given MLCP can be restated in a simpler form as

$$\begin{cases} \mathbf{A}\mathbf{x}+\mathbf{b}=\mathbf{w} \\ \mathbf{0} \le \boldsymbol{g}_U \perp \boldsymbol{\lambda}_U \ge \mathbf{0} \end{cases} \tag{23}$$

It is important to realise that the complementarity condition above is an approximation of the unilateral contact condition in equation (20). This can be seen when the vector $\boldsymbol{g}_U$ is further analyzed. This vector acts as a slack variable as part of $\mathbf{w}$ in the MLCP, and its components correspond to the approximated gap function values for all contacts in U at the next time step.

This becomes clearer when expanding $\boldsymbol{g}_U$ in equation (23):

$$\begin{aligned} &\mathbf{0} \le \boldsymbol{g}_U \perp \boldsymbol{\lambda}_U \ge \mathbf{0} \\ \Leftrightarrow\ &\mathbf{0} \le \mathbf{J}_U\mathbf{u}_+ + \boldsymbol{\varepsilon}_U\boldsymbol{\lambda}_U + \boldsymbol{\gamma}_U\frac{\boldsymbol{\phi}_U(\mathbf{q}_-)}{\Delta t} \perp \boldsymbol{\lambda}_U \ge \mathbf{0} \end{aligned} \tag{24}$$

Setting the stabilisation parameters in the matrices $\boldsymbol{\varepsilon}_U$ and $\boldsymbol{\gamma}_U$ to 0 and 1 respectively, thereby assuming a perfectly rigid contact, and multiplying through by Δt leads to

$$\mathbf{0} \le \underbrace{\mathbf{J}_U\mathbf{u}_+\Delta t + \boldsymbol{\phi}_U(\mathbf{q}_-)}_{\approx\, \boldsymbol{\phi}_U(\mathbf{q}_+)} \perp \boldsymbol{\lambda}_U \ge \mathbf{0} \tag{25}$$

where equation (14) is recovered in vector-form in the expression on the left-hand side. As we recall, equation (14) provides an approximation for the constraint function value at the next time step, which in the present case is the contact gap. This proves that the complementarity condition in equation (23) relates the approximated contact gap at the next time step to the contact's normal impulse, as required by the unilateral contact condition from equation (20).

It should be noted that the impulses of the bilateral constraints $\boldsymbol{\lambda}_B$ are unbounded, as can be seen from their respective entries in the limit vectors $\mathbf{l}$ and $\mathbf{u}$. But since all constraints are part of the same MLCP, it is straightforward to define impulse bounds also for bilateral constraints by simply specifying finite values in the limit vectors accordingly. Consequently, impulse bounds can be specified for every constraint regardless of the type. The Vortex Vehicle Systems module makes use of this feature in order to specify the maximum torque a gas engine can deliver in accordance with its characteristic torque table (see the *Engine* section of this guide). Even sticky contacts can be modeled by specifying non-zero lower bounds in the MLCP above. Accordingly, this allows you to specify an adhesive force for every contact [1], which produces bounded tensile contact forces (the colliding rigid bodies would otherwise separate). Finally, constraint impulse bounds prove very useful when it comes to modeling friction, as we'll see in the following section.

2.5.2 Friction

Dry friction arises at contact interfaces between touching rigid bodies. The resultant friction forces are dissipative and act to prevent tangential motion of the material surfaces relative to each other. For a better understanding of the friction model presented in the remainder of this section, we will define a local reference frame at a surface contact point C, as shown in Figure **5**.

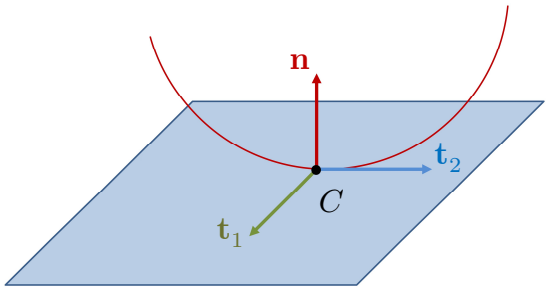

Figure 5. Local reference frame at a contact point C defined by friction axes $\mathbf{t}_1$ and $\mathbf{t}_2$, and contact normal $\mathbf{n}$.

The vector $\mathbf{n}$ defines the contact normal, and the two orthonormal vectors $\mathbf{t}_1$ and $\mathbf{t}_2$, perpendicular to $\mathbf{n}$, span the contact tangent plane. The relative linear velocity $\mathbf{v}$ between the two colliding bodies at the contact point, and the contact force $\mathbf{f}$ can be decomposed onto the local coordinate frame defined by $\mathbf{n}$, $\mathbf{t}_1$, and $\mathbf{t}_2$, and can thus be split into normal and tangent components:

$$\begin{aligned} \mathbf{f} &= f_n\mathbf{n} + \mathbf{f}_t, & \mathbf{f}_t &= f_1\mathbf{t}_1 + f_2\mathbf{t}_2 \\ \mathbf{v} &= v_n\mathbf{n} + \mathbf{v}_t, & \mathbf{v}_t &= v_1\mathbf{t}_1 + v_2\mathbf{t}_2 \end{aligned} \tag{26}$$

Commonly, the Coulomb friction model is used to define the characteristics of friction. It states two conditions. First, the friction force must act against the relative tangent velocity, $\mathbf{v}_t$, at the contact unless the velocity is zero. Second, the magnitude of the friction force vector $\mathbf{f}_t$ is bound by the magnitude of the normal force, f_n, expressed by the relation

$$\mu f_n \geq \sqrt{{f_1}^2 + {f_2}^2} \tag{27}$$

The factor μ is called the friction coefficient and is related to the friction angle θ by $\mu = \tan(\theta)$. The friction coefficient is a material property that depends on the two types of surface materials involved in the interaction, and can be specified as part of the contact material configuration [1]. The friction relation given above defines the so-called friction cone, erect at the contact point, which bounds the friction forces. The friction cone is shown in Figure **6**.

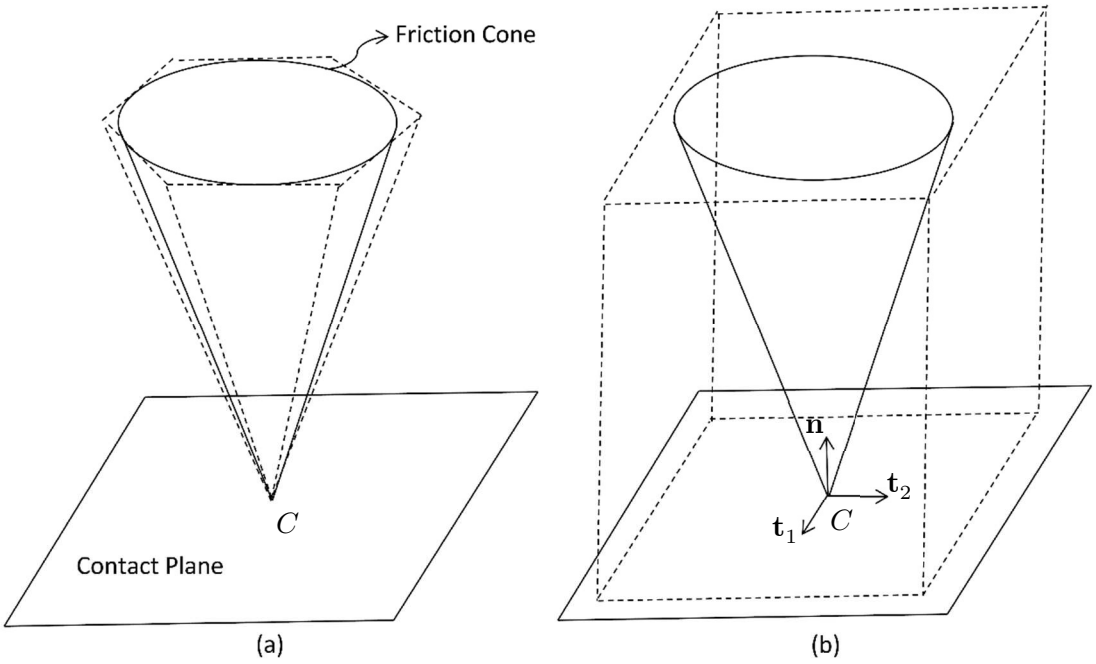

Figure 6. Friction cone: Polyhedral approximation (a) and box-approximation of the friction cone (b). The latter is used in the iterative *scaled box approach*.

Friction forces will always be inside this cone. For a sliding contact, i.e., a contact with non-zero relative tangent velocity, the friction force lies on the surface of the cone. For a static contact, i.e., a contact with zero relative tangent velocity, the friction force can lie anywhere inside the cone.

In the case of a static contact, the maximum friction force is higher than for a sliding contact. This is caused by interlocking effects of the contacting surfaces at the microscopic level and is commonly modeled as an increase of the friction coefficient which corresponds to an increase in the opening angle of the friction cone. As such, one distinguishes between static and kinetic friction with respective static and kinetic friction coefficients, μ_s and μ_k.

Assuming $\mu_s = \mu_k = \mu$ for simplicity, the Coulomb friction model, outlined above, can be expressed as

$$\begin{cases} \mu f_n \geq \sqrt{{f_1}^2 + {f_2}^2} \\ \|\mathbf{v}_t\| \left(\mu f_n - \sqrt{{f_1}^2 + {f_2}^2} \right) = 0 \\ \mathbf{f}_t \cdot \mathbf{v}_t = -\|\mathbf{f}_t\| \|\mathbf{v}_t\| \end{cases} \tag{28}$$

The friction cone condition (first line above) is non-linear, which makes its inclusion into the MLCP in equation (22) impossible. A more general, non-linear complementarity problem would have to be formulated instead, which is computationally more demanding and not feasible in an interactive simulation setting. A common alternative approach is to replace the non-linear friction cone by a polyhedral approximation as shown in Figure **6**, which yields a linear expression and leads to an MLCP. This approach is taken in the work of Anitescu and Potra [9]. However, in this case, additional unknowns and constraints need to be added to the resultant MLCP in order to model all conditions in equation (28), which increases the computational complexity of the formulated problem.

In Vortex Software, an alternative approach is taken that significantly simplifies the problem. The idea behind this approach lies in solving a sequence of MLCPs with increasingly better approximations of the friction force bounds. To this end, the friction force bounds in the MLCP, which are otherwise defined by the friction cone and are therefore dependent on the normal force, are considered as constant force limits in every iteration. Those force limits can be specified in the MLCP as part of the constant bounds vectors **l** and **u**, as demonstrated in the previous section.

Furthermore, the friction force at a given contact is represented as a linear combination of the two orthonormal tangent vectors $\mathbf{t}_1$ and $\mathbf{t}_2$, called friction axes, leading to a "box" approximation of the friction cone. This "friction box" is shown on the right side in Figure **6**. The friction box changes its size in every iteration based on the last known normal force at the contact and the friction coefficient. For every friction axis, a velocity-level constraint is added to the system which attempts to produce a zero relative velocity on the contact plane along the given axis.

In iteration $l+1$ of the procedure outlined above, an MLCP is solved that includes the set of bilateral constraints B, unilateral contact constraints U, and a set of $2|U|$ friction constraints F, with two friction constraints per unilateral contact constraint. It is defined as $MLCP(\mathbf{A}^*, \mathbf{b}^*, \mathbf{l}^*, \mathbf{u}^*, \mathbf{w}^*)$ with

$$\mathbf{A}^* = \begin{bmatrix} \widetilde{\mathbf{M}} & -\mathbf{J}_B{}^T & -\mathbf{J}_U{}^T & -\mathbf{J}_F{}^T \\ \mathbf{J}_B & \boldsymbol{\varepsilon}_B & \mathbf{0} & \mathbf{0} \\ \mathbf{J}_U & \mathbf{0} & \boldsymbol{\varepsilon}_U & \mathbf{0} \\ \mathbf{J}_F & \mathbf{0} & \mathbf{0} & \boldsymbol{\varepsilon}_F \end{bmatrix}, \mathbf{x} = \begin{bmatrix} \mathbf{u}_+ \\ \boldsymbol{\lambda}_B \\ \boldsymbol{\lambda}_U^{(l+1)} \\ \boldsymbol{\lambda}_F \end{bmatrix} \tag{29}$$

$$\mathbf{b}^* = \begin{bmatrix} \mathbf{b} \\ \mathbf{0}_{|F|} \end{bmatrix}, \mathbf{w}^* = \begin{bmatrix} \mathbf{w} \\ \boldsymbol{s}_F \end{bmatrix}$$

$$\mathbf{l}^* = \begin{bmatrix} \mathbf{l} \\ -\mu\lambda_{U,1}^{(l)} \\ -\mu\lambda_{U,1}^{(l)} \\ \vdots \\ -\mu\lambda_{U,|U|}^{(l)} \\ -\mu\lambda_{U,|U|}^{(l)} \end{bmatrix}, \mathbf{u}^* = \begin{bmatrix} \mathbf{u} \\ \mu\lambda_{U,1}^{(l)} \\ \mu\lambda_{U,1}^{(l)} \\ \vdots \\ \mu\lambda_{U,|U|}^{(l)} \\ \mu\lambda_{U,|U|}^{(l)} \end{bmatrix}$$

where $\mathbf{b}$, $\mathbf{w}$, $\mathbf{l}$ and $\mathbf{u}$ are defined as in equation (22), and $\mathbf{J}_F$ denotes the combined Jacobian matrix of all friction constraints. The term $\boldsymbol{\lambda}_U^{(l+1)}$ corresponds to the sought vector of contact impulses in iteration $l+1$. The lower and upper impulse bounds of the friction constraints in $\mathbf{l}^*$ and $\mathbf{u}^*$ are updated from the normal impulses of their associated unilateral contact constraints in the previous iteration and the friction coefficient μ, according to equation (28). Here, $\lambda_{U,i}^{(l)}$ denotes the normal impulse of the ith contact constraint in the set U in iteration l.

The vector $\boldsymbol{s}_F$ is used as a slack variable analogously to $\boldsymbol{g}_F$ in equation (22), and contains the scalar tangential contact velocities along the respective friction axes, one component per axis. It makes sure that the friction force acts always against the tangential contact velocity as required in the Coulomb friction model and as specified in equation (28). This requirement is modeled as a complementarity condition in the resultant MLCP.

The diagonal matrix $\boldsymbol{\varepsilon}_F$ is used for regularisation of the MLCP and allows us to model friction as a viscous drag force, which can be useful when simulating a slipping wheel, for example. The desired slip factor (inverse of drag coefficient) can be specified in the contact material configuration [1].

The iterative procedure is initiated by first solving $MLCP(\mathbf{A}, \mathbf{b}, \mathbf{l}, \mathbf{u}, \mathbf{w})$, which does not contain any friction constraints (see equation (22)). The normal impulses of the contacts obtained in this initial solve will be used to specify the associated friction impulse bounds for the first iteration of solving $MLCP(\mathbf{A}^*, \mathbf{b}^*, \mathbf{l}^*, \mathbf{u}^*, \mathbf{w}^*)$, as described further above. The method then proceeds iteratively.

Since during the iterative procedure the friction boxes are constantly scaled according to the last known normal impulses, the method is termed *scaled box approach* [1]. The number of friction iterations can be parameterised if higher quality solutions are required. By default, only one iteration is performed, which has been shown to give satisfactory results in a real-time simulation setting.

For staying contacts, i.e., contacts that are known from the previous time step, an alternative approach can be employed in Vortex Software. For those contacts, the friction force bounds are set according to the normal force from the previous time step. In this method, friction bounds improve over a sequence of simulation steps rather than in a single step. It avoids the iterative procedure above and is thus more efficient, while yielding comparable results in many cases. In simulations where the friction bounds are known quantities, they can also be specified explicitly by the user. For more information, please refer to [3].

2.6 Solving the MLCP

As shown in the last section, stepping the simulation forward in time requires solving an MLCP (or a sequence of MLCPs). A linear complementarity problem (LCP) is a combinatorial problem with 2^n possible solutions, where n is the number of unknowns. Therefore, enumeration methods lead necessarily to a solution if one exists. However, they are obviously not efficient and therefore inappropriate for interactive simulation.

For certain applications, such as the simulation of granular materials or deformable bodies, iterative solvers are efficient methods that produce good results [10] [11]. Iterative solvers for LCPs are commonly used for rigid body dynamics in interactive, physics-based computer games. In this domain, the motivation lies in demonstrating intriguing, visually attractive behaviour with often very little available computational time. Lack of physical accuracy is therefore only a small concern, and a very small number of solver iterations are usually enough to achieve the desired effects. Consequently, the computed solutions are not exact and inaccuracies might arise in the simulation, which are accepted.

When it comes to immersive training or real-time engineering applications, quite complex systems must often be simulated under time constraints that are equally restrictive. And in such cases, high physical accuracy is mandatory, as relying on simulations that suffer from a lack of physical correctness can cause financial losses or jeopardise operator safety. Also, very stiff constraints and high mass ratios are often involved.

Under such circumstances, iterative solvers are typically unable to deliver sufficiently accurate results in the available computational time. Only with a high number of iterations can an iterative solver converge to the exact solution in this case, or produce a solution with insignificant error.

Choosing a high iteration count, however, comes with high time consumption, which can slow down the simulation to levels below real-time or interactive update rates.

With insufficient iteration count, on the other hand, the obtained solution vector can contain significant error.

This can mean that constraints in the system would remain violated, which in particular can cause some of the following undesired effects:

- Contacts are not respected, manifesting as collision responses that are too soft.
- Friction forces are too low, causing objects to slide and never stabilise.
- Articulations, such as hinges, break down and do not remain in their sockets.
- The error in the solution vector may also vary between time steps, producing varying accelerations, which leads to jittering rigid bodies.

A lot of research has focused on reducing or preventing these artifacts when using iterative methods with low iteration counts, such as pre- and post-stabilisation of constraints or shock propagation methods [5] [12] [13]. However, the fundamental issue remains that the solution accuracy of iterative solvers is tied to the iteration count, which in turn affects time consumption.

An alternative category of LCP solvers consists of direct methods. Such solvers produce exact results that satisfy all constraints and conditions in the system. Direct solvers generally have worse time complexity than iterative solvers.

However, they are still more efficient and preferable in an interactive simulation setting where approximate solutions are not acceptable, the reasons of which have been outlined above. For an overview of direct methods for the solution of LCPs and MLCPs, see [14].

The constraint-based, velocity-level formulation in Vortex Software's multibody dynamics engine and the use of a direct solution method allow large time steps without introducing instabilities. This makes it possible to simulate complex physical mechanisms at real-time frame rates.

This is partially due to certain time step-independent operations which have to be performed in every simulation step, such as collision detection. These operations add fixed costs in terms of computational time which are independent of the chosen time step. With the ability to choose a larger time step, the time spent on these operations per second of simulated time can be drastically reduced, leading to improved efficiency.

The default time step in a Vortex Software simulation is 1/60 seconds, which corresponds to a simulation frequency of 60Hz. Obviously, with low simulation frequencies, high-frequency system

vibrations cannot be represented and are filtered out. But at the same time the system can be stepped forward a large amount of time with significantly less computational effort.

Vortex Software's multibody dynamics engine offers both a direct and an iterative solver, which can be chosen depending on the needs of the simulation.

For example, an iterative method is used for the simulation of granular material in the Earthwork Systems module (see the *Earthwork Systems* section of this guide), whereas the direct solver is used by default for the simulation of hoisting systems with taut, stiff cables and large mass ratios (see the *Cable Systems* section of this guide). Individual partitions (see the *Multibody Dynamics Simulation* section of this document) can be solved with different solver types. This makes it possible to choose the right tools for the job.

For example, one can put more emphasis on the physical accuracy in one key area of the simulation, while trading accuracy for speed in another. This flexibility makes Vortex Software appropriate for a wide range of simulations.

3 CABLE SYSTEMS

The Vortex Cable Systems module provides the capability to simulate flexible cables and complex hoisting systems. It provides out-of-the-box creation of cables, ropes and wires that are suspended over systems of winches and pulleys. Furthermore, tools are available to aid in the modeling of structures such as static pipes and pipelines, and simple cables that always remain in a catenary shape.

Specifically, the Cable Systems module supports three types of cable models:

- generic cables
- catenary cables
- pipes

These models are explained in more detail in the following sections.

At its core, the Cable Systems module makes use of the Vortex Software multibody dynamics framework presented in the previous section. It uses constraints and rigid bodies for modeling the kinematics and dynamics of cables and their physical interaction with the simulation environment. As such, cables are fully integrated with the multibody dynamics solver and other simulation components such as collision detection and fluid dynamics.

3.1 Generic Cables

A cable can be seen as a mechanical system which transfers a force from one point (e.g., a winch) to another point (e.g., a load) through an arrangement of pulleys and rings. A generic cable models such a system. It is represented by a sequence of points, such as pulleys and rings, along which a cable is passed. The length of cable between every consecutive pair of points is called a segment. Figure **7** depicts an example configuration.

3.1.1 Points and Segments

Several types of points are supported in a generic cable. An example cable which makes use of various different types is shown in Figure **8**.

The available point types are:

- **Winch:**
  A winch is defined by a rigid body which is connected to some frame of reference through a constraint that enables rotation around a specific axis, the winch axis. One such constraint is a hinge, which is readily available in the Vortex Software multibody dynamics framework. A winch can be placed at the beginning or the end of a generic cable, and is used to feed cable length into the system or retract a cable through rotation of a motorised joint. The spooling speed of

the cable depends on the rotation speed of the winch joint and the winch radius. Speed and torque characteristics of the winch are modeled using core constraint properties provided in the multibody dynamics formulation, such as the minimum and maximum torque the constraint can apply and a desired target velocity. For more details refer to the *Multibody Dynamics Simulation* section of this guide.

- **Pulley:**
  A pulley is defined by a rotating rigid body and a constraint that provides an angular degree of freedom, such as a hinge. It is used to model a mechanical pulley that directs a cable over some point in the mechanism.
- **Ring:**
  A ring is defined by a fixed point on a rigid body and a specific direction. A cable will pass through the ring and will be guided by it.
- **Attachment:**
  An attachment is defined by a fixed point on a rigid body and is used to specify the beginning or end of the cable, such as the hook in a hoisting system.

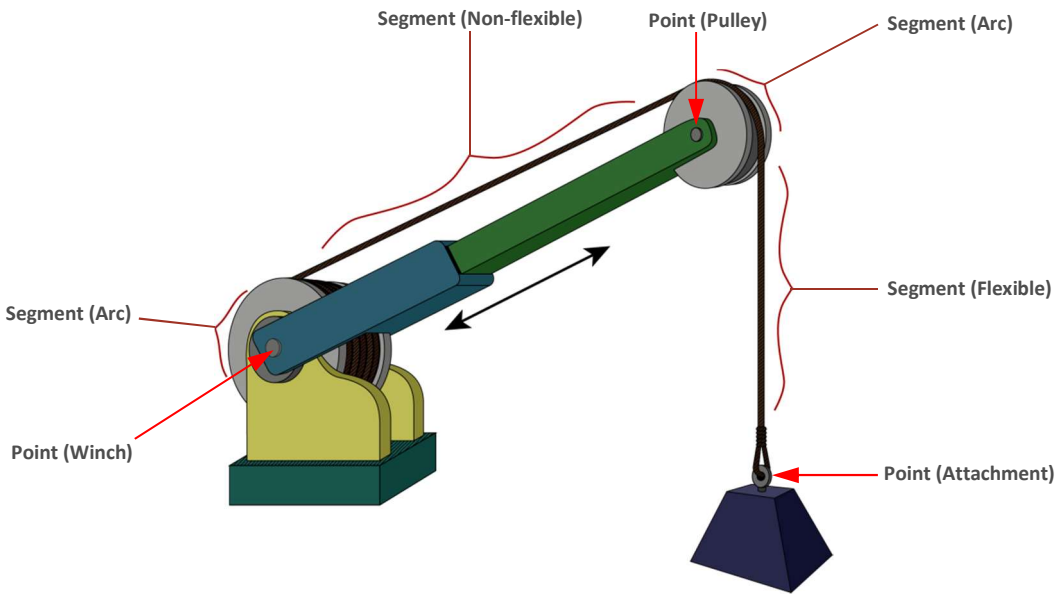

Figure 7. Points and segments: An example of a generic cable structure modeling a simple hoisting system which involves different segment and point types.

Through the constraint-based modeling approach, the dynamics of the cable and its support structure will comply with the laws of mechanics. As such, forces applied to the cable will be naturally transferred to pulleys, rings and winches, and therefore to the rest of the machine, such as a crane.

The segments in a generic cable represent the cable's geometry, its mass and mechanical properties. Depending on the required physical detail of the cable model, two different types of segments can be chosen: Flexible and non-flexible segments. These will be discussed in the following sections.

A flexible segment provides the highest level of physical fidelity in a generic cable. In flexible segments, the cable is modeled as a deformable body, which bends, twists and elongates according to solid mechanics theory. More details will be provided in the following sections.

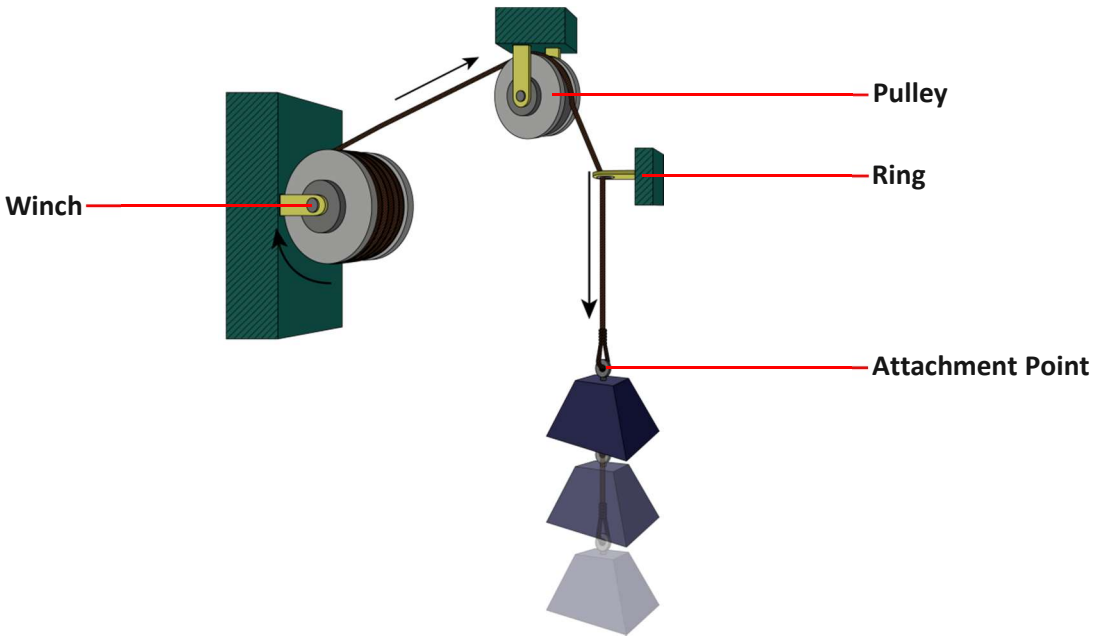

Figure 8. Generic cable configuration example. A winch is used as the first point in the cable and a load is attached to its end.

A non-flexible segment models a cable as a massless entity. It captures only the effect the cable segment has on neighboring points and segments in the generic cable, and omits any dynamics of the cable itself. It can be used as a computationally efficient tool for modeling specific operations or some specific portions of a hoisting system.

It should be noted that flexible and non-flexible segments can be freely combined in a single generic cable. They are fully coupled through the use of Vortex Software constraints and rigid bodies. Therefore, the segment type can be chosen entirely based on the level of physical accuracy at the respective location in the system. For example, in a mobile crane, the cable segments leading from the winch to the tip of the crane's boom can be modeled with non-flexible segments, thus omitting any oscillations that might arise while the cable comes under tension, whereas the final segment from the tip to the hook can be modeled with a flexible segment.

The fundamental modeling approach in a generic cable lies in choosing specific constraints to connect segments across neighboring points in the system. For example, connecting two pulleys that are a fixed distance apart with a non-flexible segment will omit the dynamics of the cable segment itself and create a constraint between the two pulley parts instead. This constraint will ensure that the rotation speed of the pulleys multiplied by their respective radii matches, and is a simplified way of representing a cable being passed from one pulley over to the next. This purely constraint-based representation of cables is referred to as massless modeling approach.

When using a flexible segment, on the other hand, the cable portion will be discretised into a certain number of (massive) rigid bodies connected by elastic constraints, modeling the elastic characteristics of the cable. The bodies at the ends of the cable are connected to the pulleys with a specialised constraint that keeps the cable tangent to the pulley surface and makes sure that it is guided by the pulley groove. An additional constraint is used to relate the rotation of the pulley to the length of the cable, feeding cable length in or retracting it whenever the pulley rotates. A winch functions in a similar way. Any flexible segment that is connected to the same pulley on the other side will be retracted when the first segment is spooled out, and vice versa. This is to ensure that the total cable length of the segments remains constant as expected.

No matter which combination of segments and point types is used, one or multiple constraints will be created between the involved bodies in order to maintain the equilibrium of forces and to correctly transfer cable length across points in the system. Some of the utilised constraints are readily available in the Vortex Software SDK, such as gear ratios or double winches. But some constraints are customised to their use in generic cables and are internal to the Cable Systems module. For more details on the constrained-based cable modeling approach, see [15].

3.1.2 Flexible Segments

As mentioned earlier, flexible segments are modeled by discretising the cable material into a series of rigid bodies connected by cylindrical joints that produce elastic bending, torsion and elongation characteristics. This representation is also known as the lumped element model, referring to the rigid bodies as lumped elements shown at the top of Figure **9**.

A cylindrical joint is modeled as six scalar constraint equations — three linear and three angular constraints that constrain the relative linear and angular motion of the connected rigid bodies respectively. The joint acts along three orthogonal axes, each axis defining a relative linear and angular motion frame for the connected rigid bodies. This situation is depicted in Figure **9**. If the cable is straight, the constraint equations are not violated, but as soon as the cable bends, twists or elongates, specific positional violations arise in the joint in specific constraint equations (see bottom of Figure **9**. More details on the parameterisation of the cable joints leading to proper elastic responses of the simulated cable are given in the next section.

Flexible segments can collide with other entities in the simulation. For this purpose, every lumped element has an associated capsule-shaped collision geometry which provides a piecewise linear discretisation of the cable shape. For improved graphical fidelity, a mesh subdivision approach in the cable visualisation is used in order to display as smooth a cable as possible.

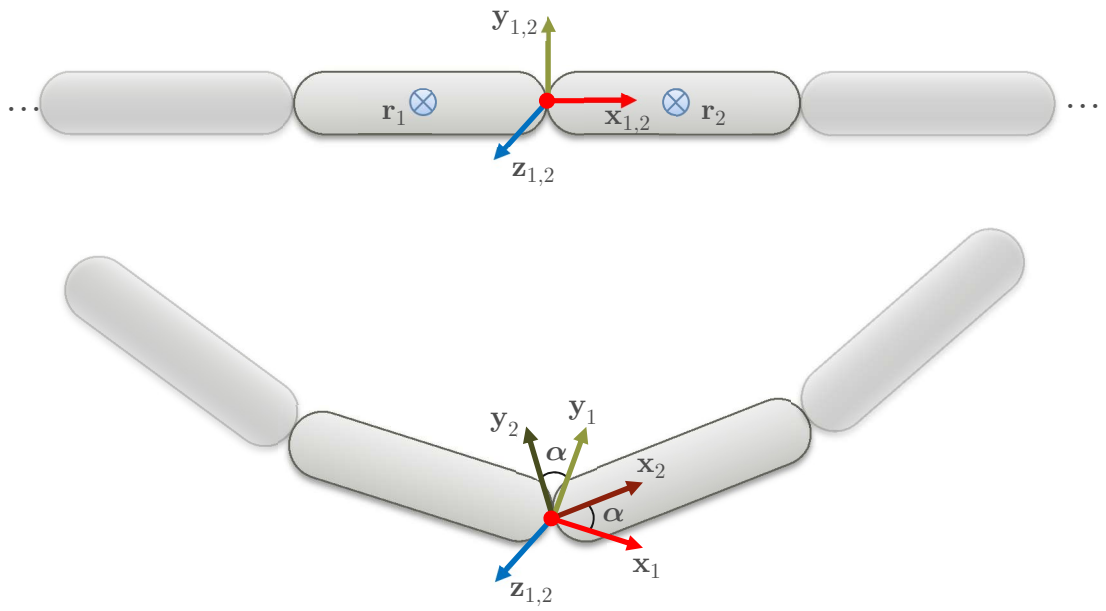

Figure 9. Cable joint model: Lumped element model of a flexible cable. Top: neighboring rigid bodies with centers of mass at $\mathbf{r}_1$ and $\mathbf{r}_2$ are connected with a joint that defines two local frames of reference defined by orthonormal vectors $\mathbf{x}_i$, $\mathbf{y}_i$ and $\mathbf{z}_i$ attached to rigid bodies $i \in \{1, 2\}$. Axes and origin of the reference frames are identical when joint is not violated, i.e., cable is straight, and not twisted or stretched at respective locations. Bottom: Cable is bent locally at joint around $\mathbf{z}$ axis, which leads to an angular joint violation α, measured in the constraint equation as a rotation between the two local frames of reference.

3.1.3 Mechanical Cable Properties

As presented in the *Stabilisation and Relaxation* section of this guide, the technique of constraint relaxation in the Vortex Software multibody dynamics framework can be used to assign elastic characteristics to every positional constraint in Vortex Software. This fact is used to model the elasticity of the cable material in accordance with established theories of elasticity. As a result, reaction forces of the cable due to stretch, twist and bending are accurately simulated.

In the Vortex Cable Systems module, the mechanical properties of a cable are set up using cable engineering terminology and concepts, which are shown in Figure **10**. The mass of the individual cable elements in the discretised, lumped element model are computed from a user-provided linear or volumetric cable density. Furthermore, elastic cable behaviour is determined from the percentage of elongation under a given load, the Young's modulus of the cable material, the cable radius, and the

number of wire threads. The elastic behaviour of a cable is captured through the use of elastic constraints at the cable joints. The appropriate elongation, bending and twist stiffness coefficients of every joint are computed from the user-provided engineering parameters mentioned above. In the following sections, this process is described in more detail. In verification experiments it was shown that the applied simulation model is accurate. For details, see the "Cables Systems" section in [16].

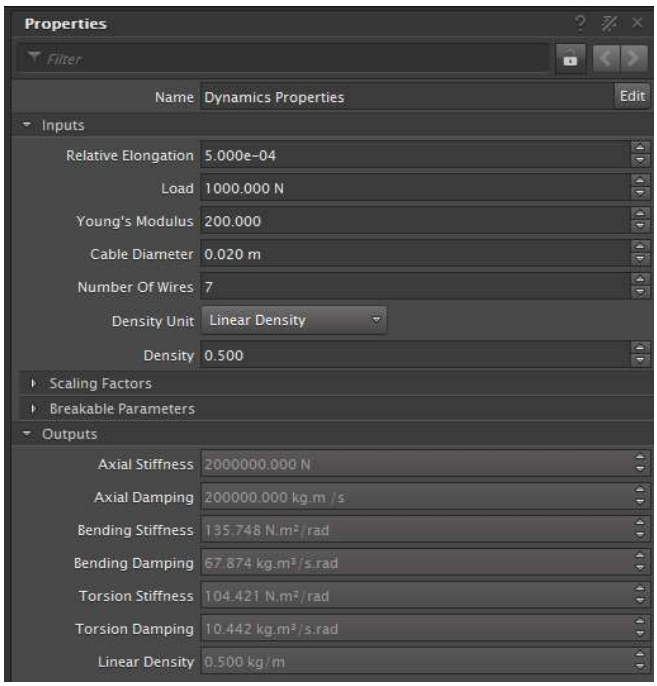

Figure 10. Mechanical cable properties: The mechanical properties used to define the behaviour of flexible cable segments under applied loads, as well as the cable's density. In the output container, the automatically computed axial, bending and torsion stiffness coefficients are shown.

Elongation

The cable's elongation (or axial) stiffness can be deduced from the user-provided percentage of elongation that should occur in the cable under a given load. Assuming Hookean material behaviour, if a load (or weight force) W is applied to a cable of length L, the cable's elongation Δl is linearly related to the material's specific axial stiffness coefficient k_a:

$$W = k_a \Delta l \tag{30}$$

Writing the relative elongation (or linear strain) as $\varepsilon = \frac{\Delta l}{\mathrm{L}}$, the expression above can be restated as

$$\frac{W}{\varepsilon} = k_a L \tag{31}$$

It should be noted that k_a is the axial stiffness of the entire length of cable. Once subdivided into a finite number of n lumped elements in the discretised cable model, the axial stiffness $k_{a,i}$ of the linear spring constraint at joint i can be computed from

$$\begin{aligned} k_{a,i} &= n \cdot k_a \\ &= \frac{W}{\varepsilon\ l_i} = \frac{K_a}{l_i} \end{aligned} \tag{32}$$

The expression above introduces the unit axial stiffness coefficient K_a, which is independent of the lengths of the individual lumped elements, and is therefore not affected by the discretisation of the cable. It is defined as $K_a := \frac{W}{\varepsilon}$ and has units of force.

Bending

Based on the Bernoulli beam theory, the Young's modulus of the cable material defines the bending stiffness of each of the cable's wire threads. According to this theory, if a beam is bent such that locally it deforms with a radius of curvature R, a restoring moment is produced which is given by:

$$M = \frac{E \cdot I}{R} \tag{33}$$

In the expression above, E denotes the Young's modulus of the cable material, and $I = \frac{\pi r^4}{2}$ is the second moment of area of a single wire's cross-section, with r being the radius of one wire thread of the cable.

The cable's lumped elements are connected by angular spring constraints, which need to have a specific bending stiffness coefficient such that the discretised cable model respects the Bernoulli beam theory. By again assuming a Hookean material, a bending moment M applied locally at some joint i (which connects two lumped elements of length l_i) induces an angular deformation θ proportional to the joint's bending stiffness $k_{b,i}$:

$$M = k_{b,i} \cdot \theta \tag{34}$$

By simple geometric analysis, the radius of curvature at joint i can be approximated by

$$R = \frac{2l_i}{\theta} \tag{35}$$

Combining equations (30) through (32), we obtain the following relation between the bending stiffness $k_{b,i}$ of the lumped elements and the Young's modulus of the material:

$$k_{b,i} = \frac{E \cdot I}{2l_i} \tag{36}$$

It should be noted that $k_{b,i}$ corresponds to the bending stiffness in a single wire thread of the cable only. The combined bending stiffness of all wire threads in the cable is computed as the sum of the contributions of the n individual wire threads. By doing so, the effect of the weaving of the threads on the effective bending stiffness in the cable is neglected.

As before, in order to obtain a stiffness term which is independent of the discretisation of the cable, we define the unit bending stiffness coefficient K_b with units of torque as

$$K_b := \frac{E \cdot I}{2} \tag{37}$$

Torsion

In order to compute the torsion stiffness of the lumped cable elements, a theory from solid mechanics is applied, which relates the restoring moment M resulting from an axial twist θ of the cable to the shear modulus G:

$$M = \frac{I \cdot G \cdot \theta}{L} \tag{38}$$

In the expression above, the term L corresponds to the cable's full length, and I denotes the second moment of area of a single wire's cross-section (see equation (33)).

As before, Hook's law is applied in order to deduce the torsion stiffness coefficient of the cable. Accordingly, an applied moment M produces a twist angle θ that is linearly related to the torsion stiffness k_t:

$$M = k_t \, \theta \tag{39}$$

Combining equations (38) and (39) and solving for k_t we obtain

$$k_t = \frac{I \cdot G}{L} \tag{40}$$

It should be noted that the term k_t corresponds to the torsion stiffness of the full length of cable. If the cable is discretised into n equidistant lumped elements, the torsion stiffness $k_{t,i}$ between two elements of length l_i can be expressed as

$$k_{t,i} = n \cdot k_t \tag{41}$$

With $L = n \cdot l_i$ and taking into account equation (40), we can rewrite this expression as

$$k_{t,i} = \frac{I \cdot G}{l_i} = \frac{K_t}{l_i} \tag{42}$$

where we defined the discretisation-independent unit torsion stiffness $K_t := I \cdot G$, which has units of torque.

Since the shear modulus G is related to the Young's modulus E and the Poisson ratio ν by $G = \frac{E}{2 \cdot (1+\nu)}$ the unit torsion stiffness can also be expressed as $K_t = \frac{I \cdot E}{3}$. In Vortex Software, a flexible segment is assumed to have a fixed Poisson ratio of 0.5. In general, the Poisson ratio can range from 0.2 to 0.5 for typical cable materials. Also, the Bernoulli beam theory is valid for cables with Poisson ratio of 0.5, as long as the cable's length is significantly larger than its radius, which is a fair assumption.

3.1.4 Advanced Simulation Techniques

As demonstrated in the "Dynamics Add-ons" section of the *Vortex Software Solution Verification and Validation* guide [16], the mechanical characteristics of flexible cable segments are accurately represented in the lumped element model. However, this approach can suffer from instabilities due to excessive transversal forces acting on the lumped elements in cases where high tensions in the cable arise. In a system without any damping, these forces would cause high-frequency vibrations, which, however, cannot be captured at the large time steps necessary for real-time simulation speeds. As a consequence, the system will blow up. This artifact can be eliminated through the use of the *internal damping* technique.

Furthermore, when using a very high cable resolution (i.e., when using many, short lumped elements) or when the cable becomes too long, the number of lumped cable elements can become excessive. This may increase the computational time in the simulation, to levels that make real-time simulation rates unfeasible. This problem is addressed by the *adaptive cable mode*.

3.1.4.1 Internal Damping

The internal damping feature allows stable simulation of flexible cables that are subject to very high tension, and which would otherwise become instable. In experiments, this feature produced stable simulation of a cable under tension for a mass ratio of 1:100000, relating the mass of a single lumped element in the cable to the mass of the attached load.

When activated, the feature increases the internal damping of the constraints in proportion to the axial tension in the cable. This means that the damping can help improve the stability of a cable under tension, which can otherwise start vibrating, but damping will not be added when a cable is slack and should be allowed to bend freely.

3.1.4.2 Adaptive Cable Mode

The *adaptive cable mode* reduces the computational time required to simulate flexible cables that have a large number of lumped elements. When activated, it constantly measures the activity of the joints in the cable, and adaptively activates or deactivates them during simulation based on the motion characteristics in the cable.

When the level of motion of a joint falls below some threshold, the joint is deactivated, and the adjacent lumped elements rigidly linked together, thus reducing the computational load. When motion is detected at the joint, it is reactivated. Joint motion is detected by performing sub-simulations of the deactivated cable joints in parallel.

The trade-off between physical accuracy and simulation speed can be parameterised by specifying a budget of joints that can be active at any point in time in the simulation. The budget is distributed over the cable segment based on the motion characteristics measured in the joints, with the goal to keep the reduction in physical accuracy to a minimum.

This adaptive cable simulation approach can lead to significant speedups in simulation time compared to the full cable simulation.

3.2 Pipes and Catenary Cables

Vortex Software offers an optimised implementation for modeling pipes and pipelines. The shape of the pipe is specified manually prior to simulation start using B-spline control points in the Vortex Editor, and

does not change after the simulation has started. The collision shape is modeled as a series of capsule collision geometries. In collision, the pipe acts as a static and infinitely massive object. An example of the use of pipes in Vortex Software is shown in Figure **11**.

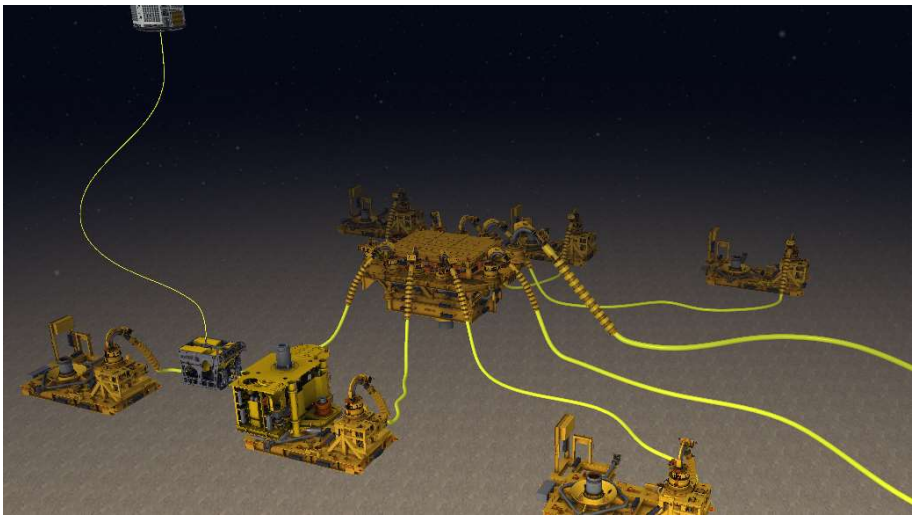

Figure 11. Subsea structure with pipes and a tethered ROV

Catenary cable models are used for the simulation of systems such as a risers and mooring lines. Unlike pipes, the shape of catenary cables can change during simulation based on the motion of a set of attachment points. However, as opposed to generic cables, their shape is animated instead of simulated dynamically. Several pre-configured catenary cable configurations are available, such as simple free-hanging catenaries and more specialised cable configurations for the modeling of subsea risers such as "lazy wave" or "steep wave" risers. As pipes, catenary cables can induce collisions with other simulation entities.

4 VEHICLE SYSTEMS

The Vortex Vehicle Systems module provides the ability to model and simulate ground vehicles integrated with the Vortex Software multibody dynamics framework. For ease of use during vehicle modeling and design, the internally employed constraint-based modeling approach is abstracted away by the user-facing Vehicle Systems data model. In this model, an extensible set of common vehicle components is provided, and can be assembled to create most common vehicle types.

4.1 Vehicle Topology

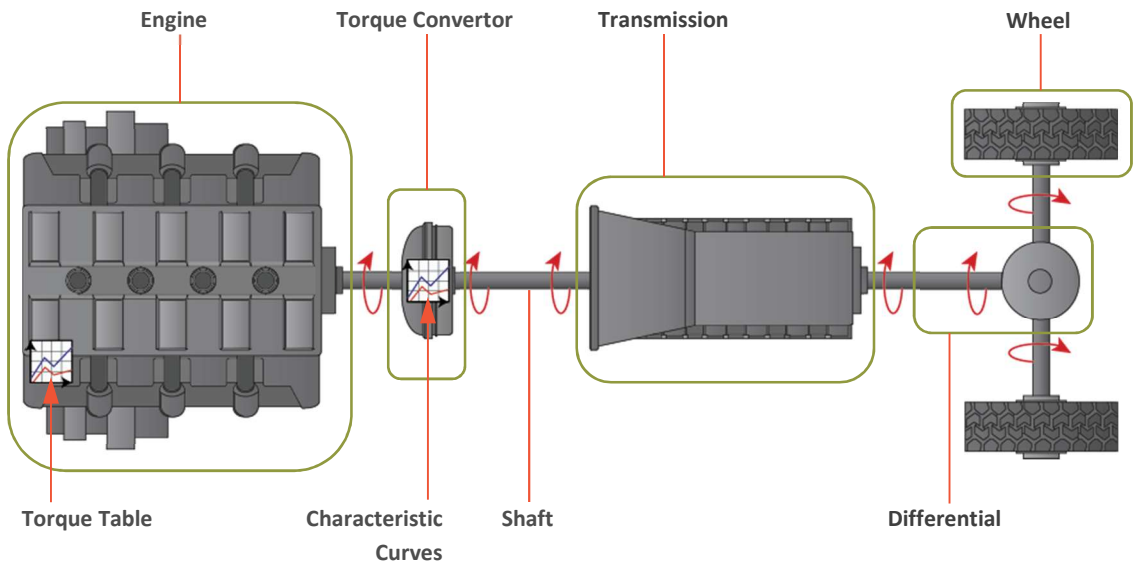

Figure 12. Vehicle components: Structure of a front-wheel-drive car powertrain. Vehicle components are emphasised in green.

A vehicle's powertrain is constructed by connecting several specific vehicle components, depending on the type of vehicle. Each component has its own set of parameters so that it can be adjusted to fit the corresponding vehicle's specifications. Examples are the maximum steering angle or the engine's characteristic torque table. The powertrain of a front wheel drive car and the components involved are shown in Figure **12**.

The topology of the vehicle component graph must correspond to a single or multiple trees (i.e., a forest). Depending on the number of inputs or outputs, a component is either the root of the tree, an inner node or a leaf. As an example, Figure **13** shows a common vehicle topology, where the *Engine* component is the root of the vehicle's powertrain, which includes a *torque converter* and a *transmission*

as inner nodes, and has a number of *wheels* as leafs. One additional trivial tree sub-structure in this example is the *chassis* component.

Internally, a compiler assembles all the independent vehicle components in a vehicle system and connects them. The vehicle topology is defined using a template that is stored in a specific file format. When the template file is loaded, a vehicle is created by adding connections between the vehicle components. Once the vehicle is created, the components can be customised using their parameterisations before starting the simulation.

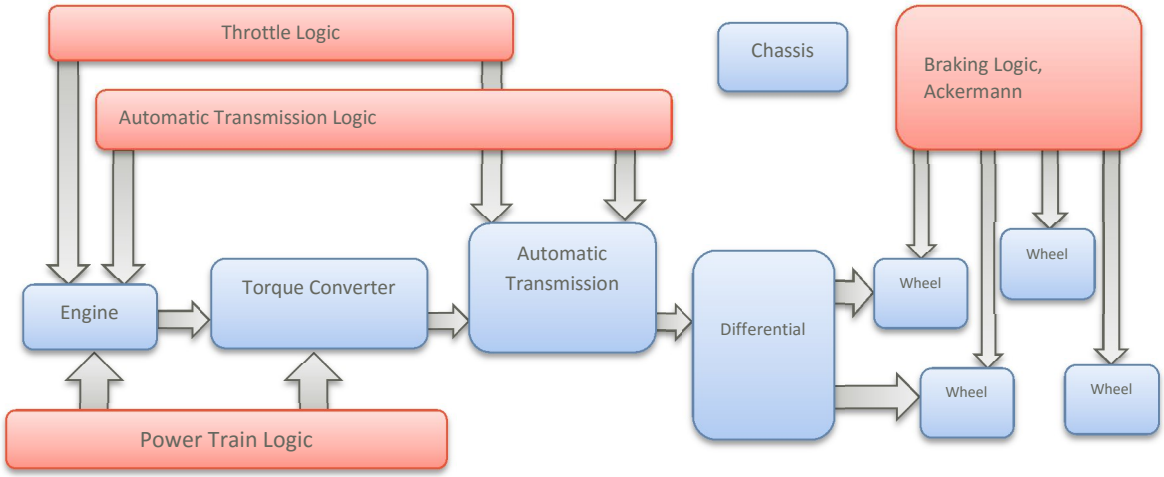

Figure 13. Vehicle topology: Topology of a front-wheel drive car, modeled with several connected vehicle components that are interacting with a logic layer.

The vehicle components are linked together through the use of rigid bodies and constraints. Most vehicle components are actually linked by physical shafts as in a real vehicle. Consequently, the dynamics of simulating the vehicle is an emerging behaviour of physically interacting vehicle components simulated using the multibody dynamics framework presented in the *Multibody Dynamics Simulation* section of this guide.

On top of the physical links between vehicle components, there is a logic layer in every vehicle, in which individual components can communicate. For instance, as shown in Figure **13** the desired throttle is communicated to the engine, which uses this information to derive the gas quantity that has to be injected. The same information is required by the automatic transmission to assess when to switch gears.

4.2 Common Components

In this section, several common vehicle components, available in the Vehicle Systems module, are presented in more detail.

4.2.1 Chassis

The chassis component contains a rigid body with mass properties (center of mass and inertia tensor) that mimic the properties of the real vehicle chassis. It also holds one or more collision geometries that represent the shape of the chassis for handling of collisions with the vehicle's environment. A vehicle can contain several chassis, which is especially useful for modeling articulated vehicles such as forwarders or wheel loaders. Another application of multiple chassis is the simulation of dependent suspension systems such as a bogie or a beam axle.

4.2.2 Engine

The engine component creates a motorised hinge joint between the chassis of the vehicle and an engine shaft, represented by a rigid body. The motorised hinge is driven according to a user-provided characteristic torque table. This table is specific to each engine and provides the maximum torque the engine can provide based on the chosen input throttle and the current engine RPM. An example torque table is shown in Figure **14**. In every simulation step, the maximum torque that the engine's hinge joint can deliver is updated according to the torque table, by setting the corresponding entries in the impulse limit vectors, $\mathbf{l}$ and $\mathbf{u}$, of the Vortex Software multibody dynamics MLCP described in detail in the *Multibody Dynamics Simulation* section of this guide.

The mass and inertia of the engine's shaft are important factors in the behaviour of an engine. Through the use of a rigid body for the modeling of the shaft, the impact of the shaft's mass properties on the vehicle dynamics is an emerging behaviour of Vortex Software's multibody dynamics formulation.

Both gas engines and electric motors are simulated using the same principal approach, that is, a characteristic torque table. However, as opposed to a gas engine, an electric motor is allowed to spin in both directions.

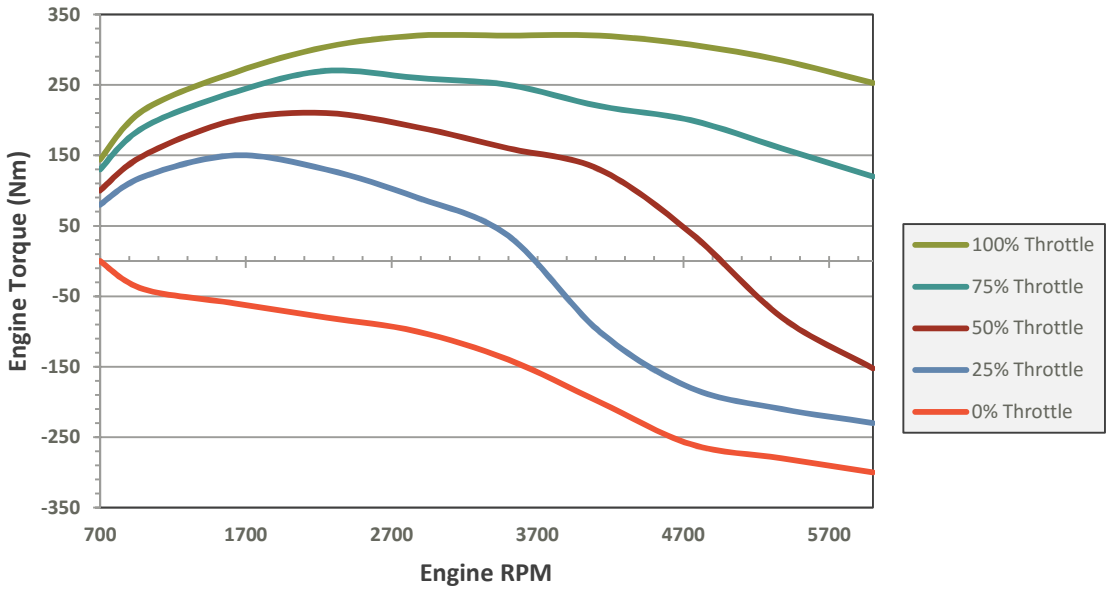

Figure 14. Gas engine torque table: Example torque table of a sports car gas engine with maximum RPM of 6000 and 320 Newton-meters maximum torque.

For a gas engine, a *start engine* logic component is introduced to simulate starting the engine. This component also models stalling when the engine's RPM falls below the level required to keep the engine running. This is very useful for simulation of a manual transmission vehicle, in which case the engine could stall when gears are shifted carelessly.

4.2.3 Torque Converter

The torque converted connects the engine to the transmission through hydraulic coupling. The torque converter component is modeled analogously to the engine, using rigid bodies and constraints. Two incoming and outgoing shafts are allowed to rotate around a fixed axis relative to the vehicle chassis. One shaft represents the propeller (or the pump) and the other one is the turbine. This setup is shown in Figure **15**.

The turbine is connected to the engine shaft, and the propeller is connected to the input shaft of the transmission (although the Vehicle Systems module allows the use of a torque converter to connect any two components). The coupling of the two shafts is modeled by a standard gear ratio constraint that forces the two shafts to rotate at the same speed. The maximum torque of the gear ratio joint is limited to be equal to the drag force that would come out of a true torque converter, which is proportional to the relative speed squared of the two shafts. For a more realistic coupling, the contribution of the

lambda curve is considered, which allows us to model the effect of the shape of the blades of the turbine and the propeller. For a detailed discussion of this subject, see [17]

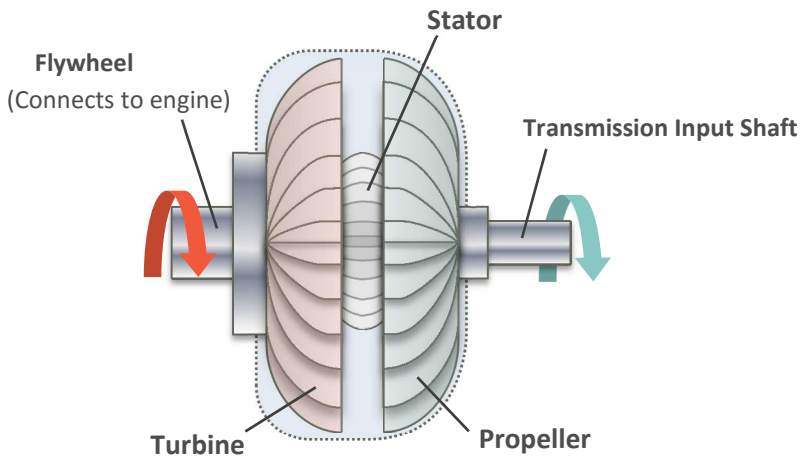

Figure 15. Torque converter: Diagram of a torque converter that connects engine and transmission through hydraulic coupling.

Another important feature modeled in the torque converter component is the torque multiplication that arises because of the special stator gear found in real torque converters. This is implemented as a lookup table that gives the torque multiplication with respect to the relative speed of the two shafts. The torque multiplication curve is provided by the torque converter manufacturer.

It is worth mentioning that the engine's characteristics and the torque converter have to be matched carefully with the real vehicle components. For instance, an engine that is too strong will rotate too fast, resulting in a less efficient system, or even cases where no torque is transmitted. If the engine is not strong enough compared to the torque converter, the engine will stall. It should be noted that the torque converter lock is not implemented directly in the component, as this feature is dependent on the vehicle manufacturer. Instead, the component exposes an interface allowing engineers to model their own locking system, which can be matched with the real system.

4.2.4 Shaft

The vehicle compiler adds shafts as required to link vehicle components. Apart from this internal shaft, a specific *shaft component* exists to which vehicle components can be linked explicitly by the user. The shaft component can have any number of outputs and thus can be used to link more than two components.

The shaft component has many applications. For instance a shaft can be used to plug a retarder component between the transmission and a differential. Another use case would be the simulation of a hybrid car. In this case a shaft can be used to link the electric motor, the gas engine and the rest of the powertrain together.

In case of a standard car, practically all of the engine's power is used to drive the wheels. However, certain vehicles use the engine's power to move other parts of the vehicle. For instance, in an excavator the hydraulics system utilises the power of the engine in order to move the mechanical arm. The shaft component provides a way to apply external torques on the engine, and as such makes it possible to simulate this scenario.

4.2.5 Transmission

4.2.5.1 Automatic Transmission

As in a real vehicle, the automatic transmission component uses the input throttle and the engine's current and maximum RPM to decide when to change gear, and to which. Multiple parameters are available to customise the transmission behaviour. As in a manual transmission, the number of reverse and forward gears and their gear ratios can be specified. Moreover it can be specified at what RPM the gear will change, the time the gear change will take, and how smoothly the gear shift will be performed.

4.2.5.2 Hydrostatic Transmission

In the Vehicle Systems module, the hydrostatic transmission is simulated by relating the angular velocities of the transmission's input and output shafts, both represented by rigid bodies, via a gear ratio joint.

In its simplest form, the constraint equation for a gear ratio joint between two bodies i and j can be written as

$$\boldsymbol{\omega}_i^T \mathbf{n}_i - r \cdot \boldsymbol{\omega}_j^T \mathbf{n}_j = 0 \qquad (43)$$

where the scalar r is the constant ratio of the joint, $\boldsymbol{\omega}_i$and $\boldsymbol{\omega}_j$ denote the angular velocities of the bodies, and $\mathbf{n}_i$ and $\mathbf{n}_j$ are some specific axes of interest. It can be seen that the gear ratio joint constrains the angular velocities of the two bodies around the given axes to respect the desired ratio r.

With the constraint equation of a gear ratio joint in mind, we can now proceed to derive the ratio r required to model a hydrostatic transmission. A hydrostatic transmission uses a hydrostatic pump that converts the input shaft speed (e.g., the engine's shaft) into pressured fluid in a pipe. At the input shaft, the flow Q_{in}, with units of m^3/s, can be written as

$$Q_{in} = V_{in} \cdot \frac{\omega_{in}}{2\pi} \tag{44}$$

where V_{in} is the stroked volume in m^3 per shaft turn and ω_{in} denotes the angular speed of the input shaft in rad/s. This equation assumes an ideal pump with 100% efficiency.

Equation (44) holds also at the output shaft of the transmission where the flow Q_{out} is converted back to angular speed:

$$Q_{out} = V_{out} \cdot \frac{\omega_{out}}{2\pi} \tag{45}$$

Under the assumption that the pressure is conserved in the pump, we can write

$$\begin{aligned} Q_{in} - Q_{out} &= 0 \\ \Leftrightarrow\ V_{in} \cdot \frac{\omega_{in}}{2\pi} - V_{out} \cdot \frac{\omega_{out}}{2\pi} &= 0 \\ \Leftrightarrow\ \omega_{in} - (V_{out}/V_{in}) \cdot \omega_{out} &= 0 \end{aligned} \tag{46}$$

It can be seen that the last row in the equation above corresponds to the constraint equation of the gear ratio joint given in equation (43). Therefore, in order to model the hydrostatic transmission using a gear ratio joint, the ratio must be set to $r = V_{out}/V_{in}$. Note that r may be variable for many drive systems.

The hydrostatic transmission component in the Vehicle Systems module has only one input shaft but can have as many output shafts as required. An output shaft can be used to power wheel or track components, but can also be employed to power any rigid body, such as the piston rod of a hydraulic cylinder actuating an excavator bucket.

4.2.5.3 Manual Transmission

The manual transmission component allows the driver to change gears manually. It also detects if the driver succeeds in changing gear, and reports this information.

Most drivers press the clutch to change gears. This is required to successfully start a vehicle in any gear except neutral without stalling the vehicle, but it is not mandatory for shifting gears. Advanced drivers can shift the gear without pressing the clutch. In a real vehicle the gear shifting is successful if the input shaft angular speed of the manual transmission is close to the angular speed of its output. In the Vehicle Systems module, as in a real vehicle, the manual transmission "doesn't know" if the clutch is engaged or not. It is purely the mechanical behaviour of the interconnection between the vehicle components involved that makes shifting a success or not. In pressing the clutch while shifting gears, the shaft between the clutch and the manual transmission is free. This allows the shaft to change speed easily,

which facilitates shifting gears. In general, a clutch component allows simulation of the connection between two shafts that can be progressively engaged or disengaged. It can therefore also be used outside the manual transmission. Also, the maximum torque of the clutch can be parameterised to simulate different clutch types.

4.2.6 Differential

The differential component has one input shaft and two output shafts. It ensures that the following equation is always respected:

$$r_{in} \cdot \omega_{in} = r_1 \cdot \omega_1 + \ r_2 \cdot \omega_2 \tag{47}$$

where ω_{in}, ω_1 and ω_2 denote the angular speeds the input shaft and the two output shafts respectively, and r_{in}, r_1 and r_2 are the desired corresponding speed factors.

In order to simulate an *open differential*, the factors are set to $r_{in} = 1$, $r_1 = -0.5$ and $r_2 = -0.5$ such that the speed at the input shaft is equally distributed to the two output shafts.

For a *locking differential*, an additional gear ratio joint is added which locks the two output shafts to an identical speed by using a ratio of $r = -1$ in the gear ratio constraint equation given in equation (43)

Finally, a maximum torque can be specified in the gear ratio, which models the *locking differential* using the concept of Lagrange multiplier limits introduced in the *Friction* section of this guide. This simulates a *limited slip differential*, which allows one wheel to spin slower than the other if a sufficiently high external load is applied on it. This can be caused by friction forces acting between the wheel and the ground.

4.2.7 Steering Systems

The Vehicle Systems module offers several steering modes that can be selected by the driver in an ongoing simulation. The different systems are presented in this section.

4.2.7.1 Ackermann steering

The Ackermann steering model makes sure all wheels follow the vehicle's trajectory while turning. In other words, they all turn around a common point O shown in Figure **16**. This avoids excessive lateral friction forces that could damage the tire or lead to instabilities while turning.

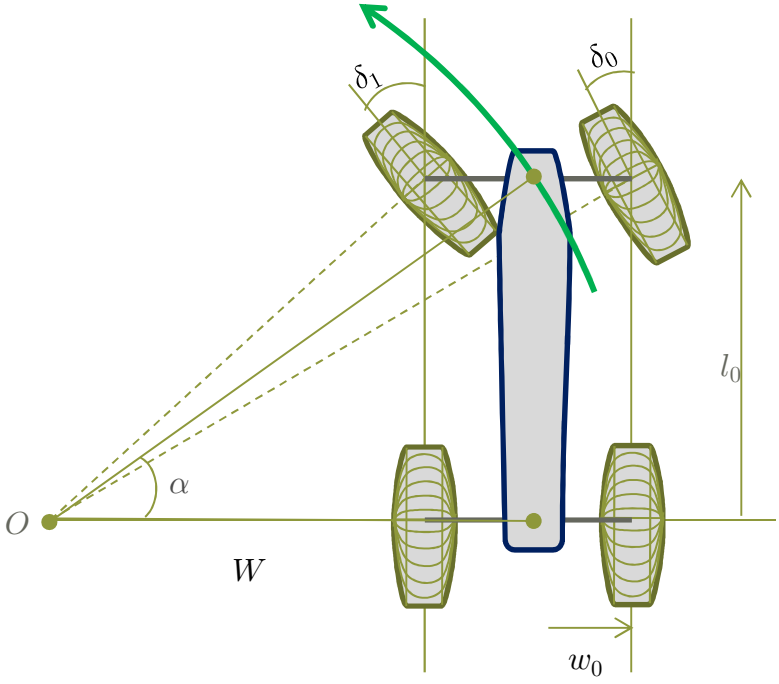

Figure 16. Ackermann: A left turn in the Ackermann steering model.

The Ackermann steering component has a normalised steering input s in $[-1,1]$, which is used to compute the desired steering angle α, from the maximum steering angle α_{max} as

$$\alpha = s \cdot \alpha_{max} \tag{48}$$

The steering angle δ_i of the ith wheel is computed from its local longitudinal coordinate l_i, and its lateral coordinate w_i, shown in Figure **16** using the expression

$$\delta_i = \text{atan}(\frac{l_i}{W + w_i}) \tag{49}$$

with $W = l_i/\tan(\alpha)$.

4.2.7.2 Pivot and Crab steering

The pivot steering mode allows the vehicle to turn around a fixed pivot point by setting the angle of wheel i with local coordinates given by l_i and w_i, depicted on the left side of Figure **17**, to $\delta_i =$

$\text{atan}\left(\frac{l_i}{w_i}\right)$. In crab steering mode, all wheels are set to the same angle δ as shown on the right side of Figure **17**.

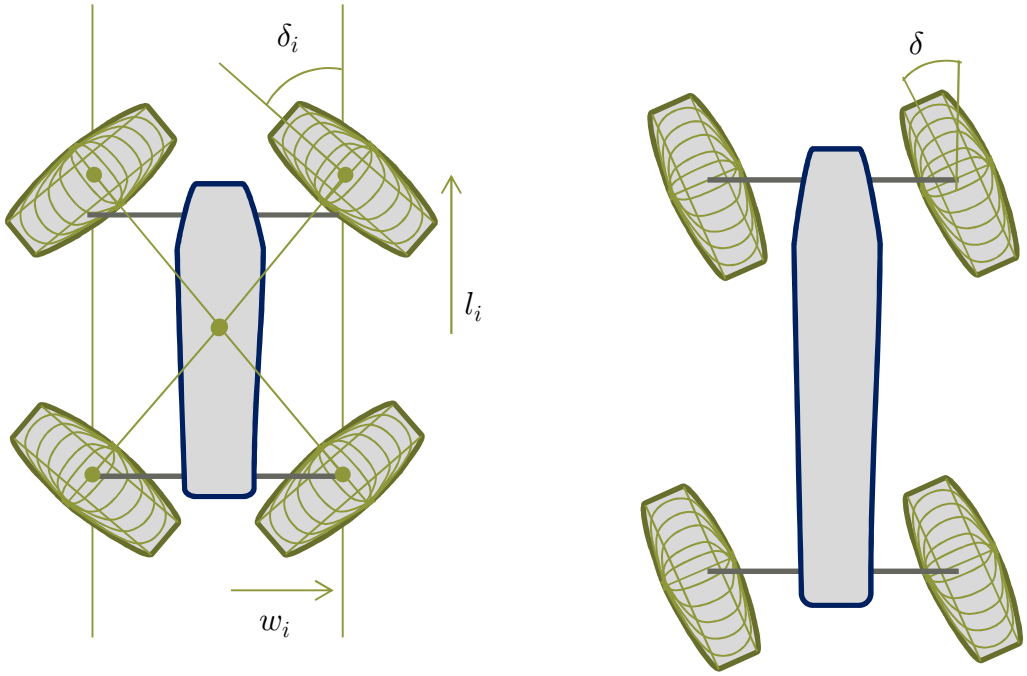

Figure 17. Illustrations of pivot (left) and crab (right) steering mode.

4.2.8 Wheels and Suspension

The Vehicle Systems module allows configuring a vehicle with any number of wheels. Their point of attachment can be anywhere relative to the chassis. By default, wheels are attached directly to the vehicle chassis using the car wheel joint for wheeled vehicles and the double hinge joint for vehicles with flexible tracks. These joints are natively provided by the Vortex Software multibody dynamics engine. Both can constrain all six relative degrees of freedom (DOF) between a wheel and the chassis.

It should be noted that apart from the built-in suspension types presented in this section, custom suspensions are supported in Vehicle Systems. A wide variety of suspension systems can be modeled simply by using additional rigid bodies and joints, and instead of attaching the wheels directly to the vehicle chassis, connecting them to an intermediate chassis component which could represent a hub carrier or a bogie.

The *car wheel joint*, which represents a simplified, idealised suspension system, is shown on the left in Figure **18**. Its controllable coordinates are rotation about the wheel's rotation axis, a displacement along the suspension axis and a rotation about the steering axis. Depending on the powertrain topology, the wheel's rotating axis can be free or driven by the powertrain. The suspension coordinate is locked at a

given position, and the corresponding constraint equation is relaxed to obtain characteristics of a spring-damper, by using the technique described in the *Stabilisation and Relaxation* section of this guide. The stiffness and damping coefficient can be chosen according to the characteristics of the suspension in the real vehicle. By default, the steering coordinate is driven by the Ackermann steering component presented in the last section, but it can be replaced by any other control logic.

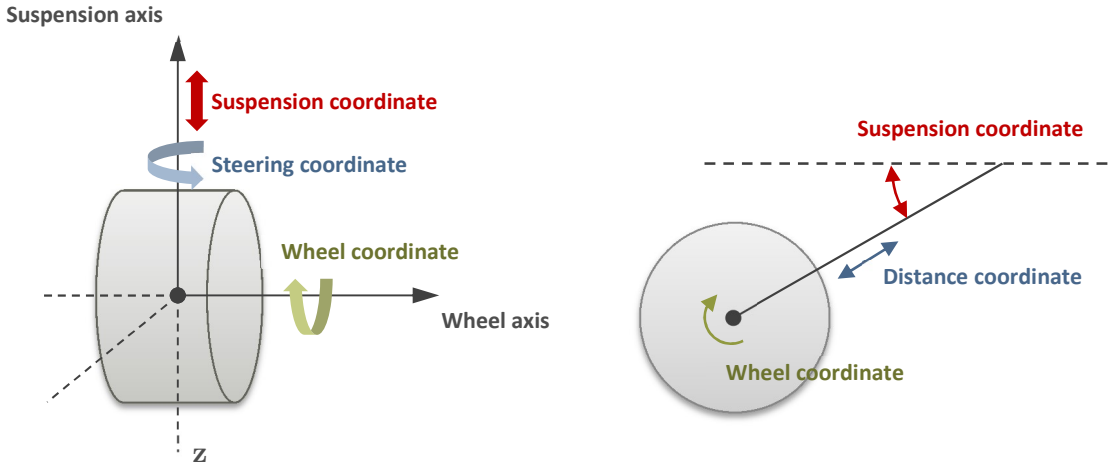

Figure 18. Suspension joints: Joints representing the built-in suspension types in the Vehicle Systems module. Left: Car wheel joint with linear suspension coordinate. Right: Double hinge joint with angular suspension coordinate.

The controllable coordinates of the *double hinge joint*, shown on the right in Figure **18**, are a rotation about the wheel's rotation axis, a rotation about the suspension axis and the distance between the joint's attachment point on the chassis and the wheel center. When modeling a flexible track, the wheel rotation axis is free, as its rotation is driven by the track's movement. The suspension coordinate is locked at a given angle and, as in the case of the car wheel joint, the corresponding constraint equation is relaxed to obtain visco-elastic characteristics. The stiffness and damping coefficients are chosen according to the torsion bar suspension of the real vehicle. The distance coordinate is locked without any relaxation of the constraint equation in order to model a rigid suspension lever.

4.3 Wheel-Ground Interaction

During simulation, the interaction between a wheel and the ground is modeled in Vortex Software using one of several approaches, depending on the nature of the interaction. The Vehicle Systems module provides a rich set of wheel-ground interaction models, referred to as *tire models*, which can be divided into two categories: Hard ground and soft ground tire models.

For example, soft ground tire models account for increases in rolling resistance and non-linear changes in friction forces as the wheel sinks into soft ground, as shown in Figure **19**. Furthermore, the effects of tire pressure are considered by both hard and soft ground tire models.

The theoretical basis of the employed tire models has been developed over several decades in the fields of terramechanics and vehicle dynamics, and the models have been verified in extensive field experiments [18]. Vortex Software integrates such models, which for the most part are based on constitutive equations, into the constrained-based modeling framework presented in the *Multibody Dynamics Simulation* section of this guide, and thus makes them available in a real-time, dynamics simulation context. For technical details on the integration and the modeling approach, see [19].

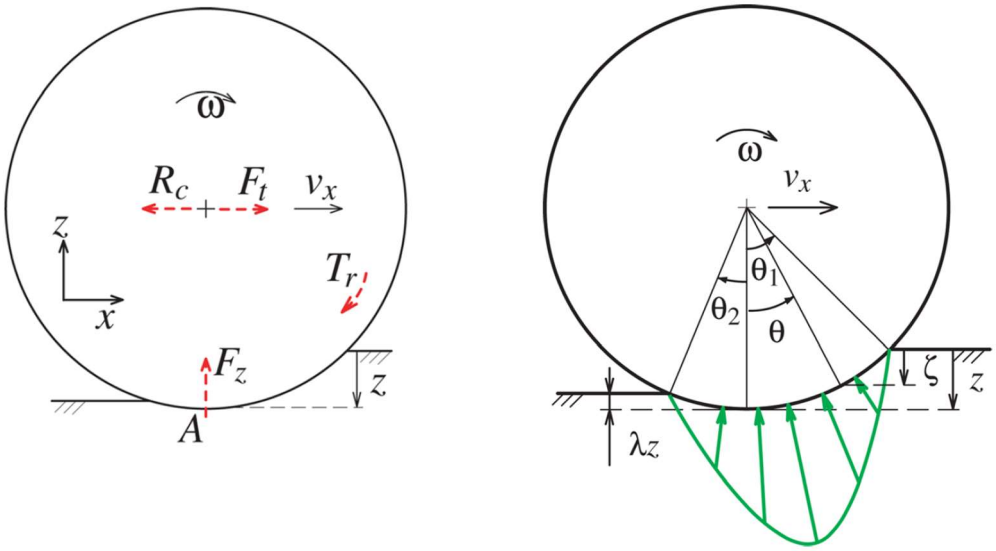

Figure 19. Left: Soil reaction forces and torques applied to a rigid wheel moving over soft soil. Right: Normal stress distribution (illustrated in green) assumed by the Wong-Reece model. Reproduced with permission from authors of **[19]**.

4.3.1.1 Hard Ground

Hard ground tire models compute the friction force bounds and alignment moment that arise at the wheel/ground interface on hard ground, and thus capture the non-linearity in longitudinal and lateral friction. Hard ground tire models supported in Vortex Software are the magic formula [20], Fiala [21], and composite slip [22] models.

4.3.1.2 Soft Ground

Soft ground tire models capture the non-linear friction behaviour and rolling resistance resulting from wheels sinking into the ground, in order to produce the correct drawbar pull of the vehicle. These models can be categorised according to the specific constitutive equations they use to relate the stress and strain acting in the ground. Two classes of stress-strain relations are defined, which depend on the

direction of the stress applied in the ground, i.e., normal or tangential to the wheel surface. As such, two different classes of constitutive equations exist, one describing the normal stress-strain relation (pressure-sinkage relation) and the other describing the shear stress-strain relation (shearing relation) [23] [24]. The supported pressure-sinkage relations in Vortex are Bekker, Bekker-Wong, Reece, muskeg, and snow. The supported shearing relations are Wong, exponential and hump [18]. An example of the normal stress distribution assumed in a soft ground tire model is shown in Figure **19**.

5 EARTHWORK SYSTEMS

The Vortex Earthwork Systems module is tailored to the needs of high-fidelity, real-time earth-moving simulations, and employs physically-based soil deformation algorithms. Terrain deformations, caused by earth-moving tools such as buckets and blades, and the corresponding terrain reaction forces, are captured in real time and full two-way force coupling with other simulation entities is modeled. The interactions between machine and soil are fully simulated, allowing soil to be cut, compressed and spilled, all inside an interactive environment.

5.1 Simulation Method

For the interaction of tool and ground, the Earthwork Systems module employs a hybrid particle-based and mesh-based soil simulation method originally presented in [25] and [26] and later extended by incorporating soil reaction forces acting on cutting tools (computed based on a validated semi-empirical model [27]). The system adaptively switches between particle and mesh representations of soil based on the simulation context, which makes the method very efficient. This approach is motivated by the occurrence of soil in nature: In equilibrium and undisturbed by outer forces, it remains in a static state, not showing any movement, while once exposed to a sufficiently high stress, soil shows high plasticity. Thus, highly dynamic phenomena can be simulated in an efficient way. The method combines a volumetric grid representation of soil at equilibrium with a discrete element method (DEM) particle simulation to model soil in motion. The grid representation is based on a height field combined with a layered data structure of rectangular soil columns shown in Figure **20**.

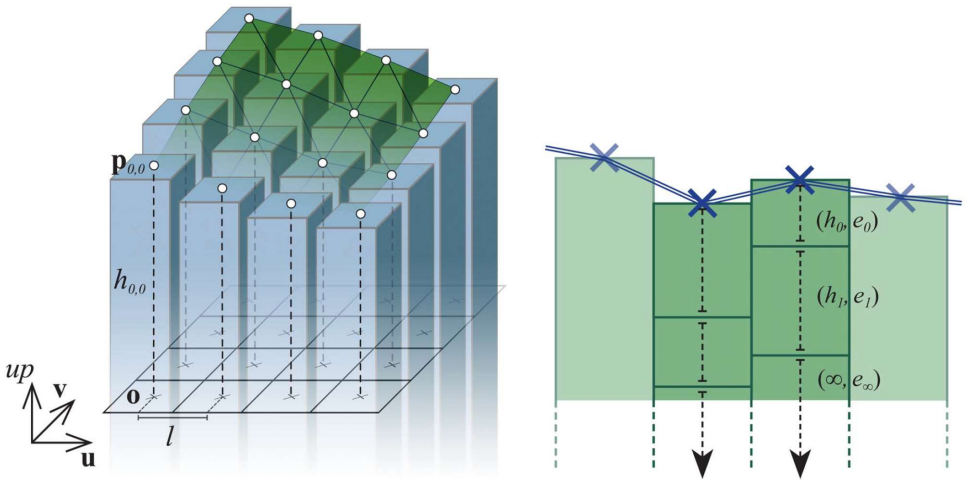

Figure 20. Left: Height field-based soil grid. Model of a continuous soil body as 2-dimensional array of rectangular soil columns (blue) and its visualisation (green). Right: Representation of soil columns as linked lists of soil layers. To store the soil's compaction degree in the soil grid each entry consists of a height and a void ratio value $(\boldsymbol{h}, \boldsymbol{e})$.

5.1.1 Soil Cutting Forces

In this hybrid setting, the reaction force applied to cutting tools is based on two combined models. First, when horizontal motions of a tool are detected in the grid, such as a blade carving through the ground, DEM particles are introduced adaptively, replacing portions of the grid in the vicinity of the tool, as shown in Figure **21**. These particles represent the excavated and disturbed soil and form the surcharge material, which is accumulated in or in front of the tool. Forces between particles and tool are exchanged through two-way force coupling. This way, the reaction force applied to the blade correctly increases with increasing surcharge.

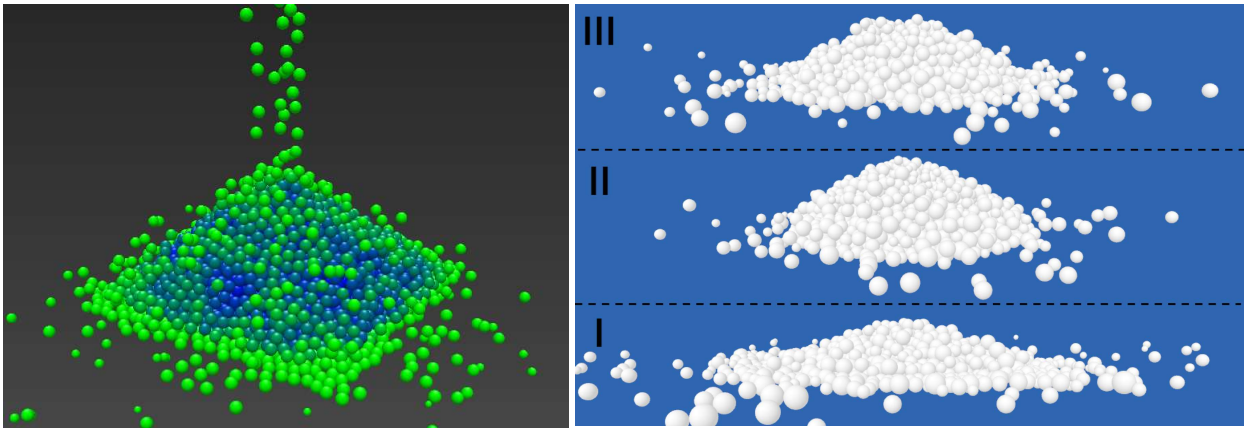

Figure 21. Parallel particles (P^2) simulation results. Left: Loosely poured, stable pile of soil particles with compactness visualisation (green for low and blue for high compaction degree). Right: Different realistic angles of repose achieved with varying kinetic friction angle μ_k and adhesion a. I: $\mu_k = 5, a = 0$ Pa. II: $\mu_k = 30, a = 0$ Pa. III: $\mu_k = 5, a = 1000$ Pa

The DEM method is based on Parallel Particles (P^2), a technique for the simulation of granular materials that was developed especially for the purpose of real-time simulation of soil-cutting operations in high detail [28]. A parallel solver leads to very high simulation speeds. The method uses intuitive physical contact properties such as stiffness, damping and friction coefficients, and leads to physically plausible simulation results, such as stable soil piles with realistic angles of repose for various material types (see Figure **21**).

Secondly, the portion of the terrain in front of the tool that has not yet undergone failure (represented in the grid) requires the tool to apply a certain minimum force in order to fail, which subsequently leads to soil displacement. This force is computed via a semi-empirical model and must be overcome by the tool in order to progress. A detailed contact analysis between tool and grid is performed to detect if the tool is displacing soil, in which case a corresponding part of the grid is replaced by particles.

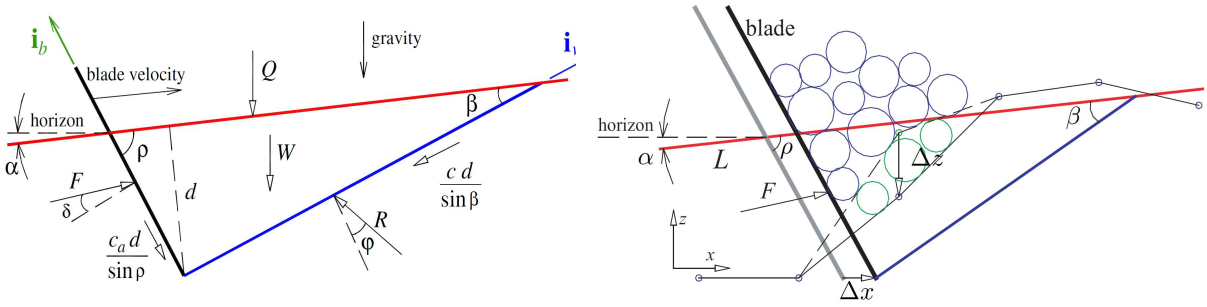

Figure 22. Fundamental earth-moving equation (FEE) cutting force model (left) with geometric and physical input parameters in combination with adaptive height field deformation algorithm (right). The height field vertices (blue/green dots) serve as input for the soil slope approximation (*L*). The blade motion Δx causes height field deformation Δz and particle generation (green circles). Particles from previous simulation steps (disturbed soil) are depicted as blue circles. Tool needs to overcome soil reaction force F in order to progress and cause soil deformation (Δz) and generation of particles (green circles).

This process is shown in Figure **22**. No soil displacement is allowed if the tool does not apply sufficient force. The force required for soil failure to occur is computed through a formulation obtained via the fundamental earth-moving equation (FEE) of Reece [29] and the method of trial wedges presented by McKyes [30].

As shown in Figure **22**, the soil reaction forces computed via the FEE formulation depend on various geometric parameters such as the tool's angle to the ground *rho* or the tool's penetration depth *d*, and on the soil's shear strength parameters internal friction angle *phi* and cohesion *c*, which are readily obtained from the grid. The soil volume in front of the tool is discretised in multiple slices as explained in [27] , and for each slice the required geometric parameters are computed through a tool/grid contact analysis. In this discretisation approach, each slice leads to a separate reaction force, which has the effect that plausible reaction forces are calculated regardless of the soil's surface shape and inclination in front of the tool. All forces are integrated and added to the tool via a 6-DOF constraint (see the *Constraints* section of this guide).

In reality, the surcharge material that accumulates in front of a cutting tool means that it is harder for the unfailed soil in front of the tool harder to fail, i.e., higher tool forces are required for soil failure to occur. This effect is also captured in the simulation, by feeding forces from the surcharge particles back into the surcharge force parameter *Q* of the FEE model. This procedure is depicted in Figure **23**. The soil reaction force is further increased by the accumulation of surcharge particles, which themselves apply contact forces to the tool. Consequently, during simulation, a bulldozer can correctly become stalled, if, with increasing surcharge, the total soil reaction force overcomes the bulldozer's traction force.

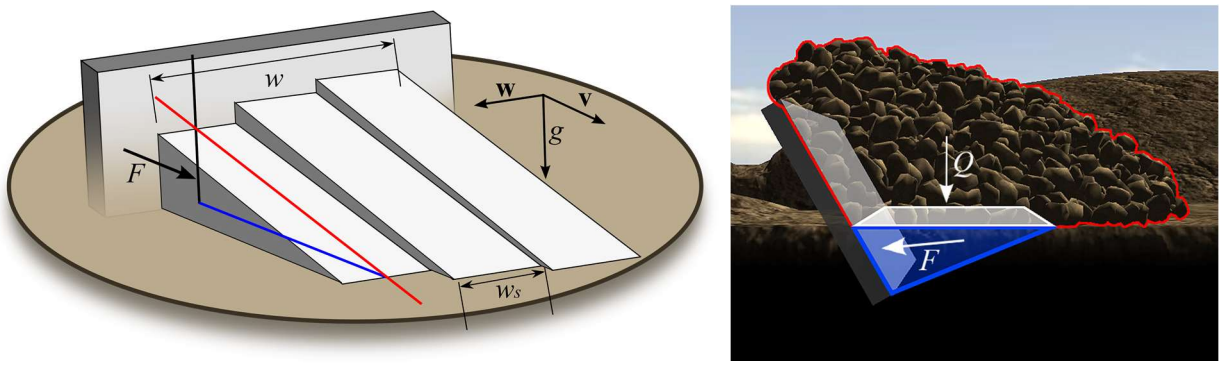

Figure 23. Left: Soil in front of tool is discretised into several soil slices, each contributing its own soil reaction force. Right: Surcharge force Q caused by soil particles traveling on soil wedge further increases the soil reaction force of the undisturbed soil in front of the tool.

Formulating the soil-cutting force based on the FEE is motivated by the fact that (i) it is a 2D model, which makes it easy to apply and computationally efficient, and (ii) it has a reasonable range of operations appropriate for the simulation of several soil-cutting tools. As mentioned by Shen and Kushwaha [31], this equation is approximately valid for tools that have a sufficiently high width-to-depth ratio. This allows us to use the FEE formulation for wide tools, such as bulldozer blades and wheel loader buckets. Narrow tools, by contrast, not only induce horizontal and vertical soil movement, but also movement to the sides, in direction of the tool width. In this case, the FEE is not valid anymore. However any tool with side walls can be simulated by the FEE reasonably well since the side walls restrict sideward motion of the soil. This justifies application of the FEE model to buckets of earth-moving excavators and backhoes.

5.1.2 Shear Strength and Compressibility

Soil models are parameterised using internal friction angle and cohesion, which are established shear strength parameters from the field of soil mechanics. Since shear strength can change with compactness [32], the soil's compression state is tracked throughout the simulation. A compaction model is used to simulate elasto-plastic deformations that occur in the grid due to vertical stresses applied to the medium by cutting tools. In this model, the volume change of soil due to compaction — the stress-strain relationship — is described by an empirical model from the field of critical state soil mechanics [33].

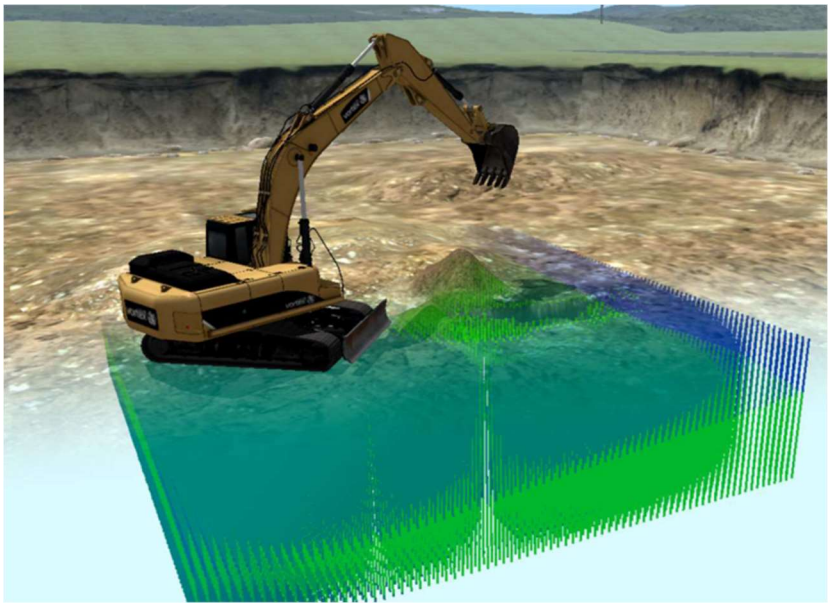

Figure 24. Soil compaction profile inside an earthwork zone. Blue denotes a high degree of compaction, while green denotes a low degree of compaction.

As a result of compaction, the shear strength parameters in the grid change locally in the vicinity of the interaction, which, depending on the soil type, makes the soil harder to fail and compact at this location [32]. In this way, the history of earth-moving operations is considered during the simulation. This greatly increases realism compared to methods that treat soil as homogeneous. For example, during trenching, soil can initially be in a compacted state, and therefore relatively high cutting forces must be applied for it to fail. Once the soil is excavated, it can loosen up, leading to reduced compactness and thus shear strength. When it is piled up beside the trench, in our method this information is kept and stored in the grid as shown on in Figure **24**. Consequently, during backfilling, a lower cutting force is required to move the excavated soil back into the trench.

Apart from the cutting forces, the shear strength parameters also define at which surface angles soil starts to slip. We perform a stability analysis in the grid based on the Mohr-Coulomb failure criterion, which models the resistance of soil to shear stress. Once failure occurs, slip is modeled by computing soil restoring forces and from those integrating flow velocities on the grid. The stability analysis and the Mohr-Coulomb criterion are briefly depicted in Figure **25**. More details can be found in [25] and [26].

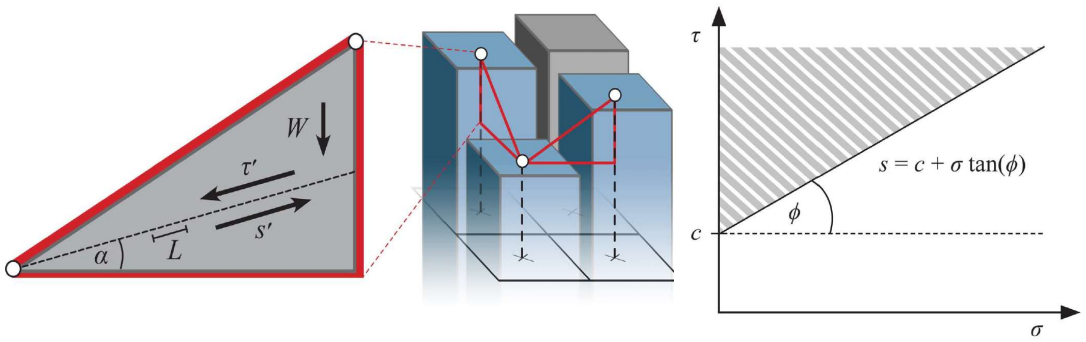

Figure 25. Left: Soil stability analysis in the discrete soil grid. Soil with weight force W fails along failure line L at angle α to horizon, which is the angle where shear strength force s' equals shear stress force τ'. Right: Mohr-Coulomb criterion: yield line with slope $\tan(\phi)$ and intercept c for internal friction angle ϕ and cohesion c describing the linear relationship between shear strength s (resisting an applied shear stress τ) and normal stress τ.

5.2 Simulation Results

To date, earth-moving simulations performed using the Earthwork Systems module have proven satisfactory in several areas and have been validated by actual operators [27]. As a consequence of the physically-based modeling approach, the applied method captures emerging behaviour of soil naturally, as opposed to approaches based only on animation. Simulations have given physically plausible results, running at interactive–to–real-time rates. The experiments showed several emerging effects, such as a bulldozer's travel speed and trajectory being affected with increase in cutting force (see Figure **26**). These effects are hard or impossible to capture accurately without the use of physical models. Simulation results can be found in [34] and [35].

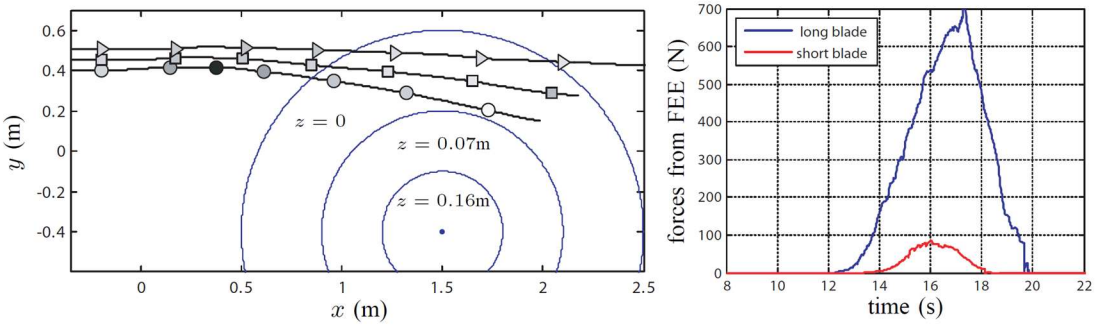

Figure 26. Experiment: Simulation of vehicle with mounted bulldozer blade driving into a concentric pile of soil. Left: Trajectories of the vehicle from three simulation runs with varying blade lengths ranging from short to long. Height of pile indicated with iso-lines. Right: Blade cutting forces applied by FEE model with long and short blade. More details in **[34]** and **[35]**.

6 FLUID INTERACTION

Vortex Software supports the real-time simulation of dynamic rigid bodies immersed in fluid, and simulates several hydrodynamics effects that are a result of the interaction between the rigid body and the surrounding fluid. It should be noted that the fluid simulation capabilities offered by Vortex Software do not provide the same level of physical accuracy as complex computational fluid dynamics (CFD) methods.

Instead, approximate fluid forces are computed using computationally efficient methods, which makes the approach well-suited for use in real-time, immersive simulation environments.

A fluid in Vortex Software is represented by a surface, such as a plane, a height field or a general mesh. A rigid body that is overlapping with or fully below the fluid surface will be subject to hydrodynamics forces explained in the following sections.

Buoyancy, drag and lift are simulated based on constitutive equations, and their effect is added to the submerged rigid body as generalised forces, i.e., forces and torques. This process is shown in Figure **27**. The computed generalised forces are incorporated into the Vortex Software multibody dynamics formulation through the $\mathbf{g}$ term in equation (4), given in the *Multibody Dynamics Simulation* section of this guide.

Fluid forces are calculated according to the shapes of the collision geometries attached to the rigid body, their immersed portions in the fluid, and the velocity of the fluid relative to the body's surface. Vortex Software offers fluid interaction with various geometric primitives (boxes, spheres, capsules, cylinders), as well as convex and general meshes.

For general meshes, fluid forces are computed through an integration process over all surface triangles. Consequently, with higher mesh resolution, the computed fluid forces are more accurate, while the computational cost is higher.

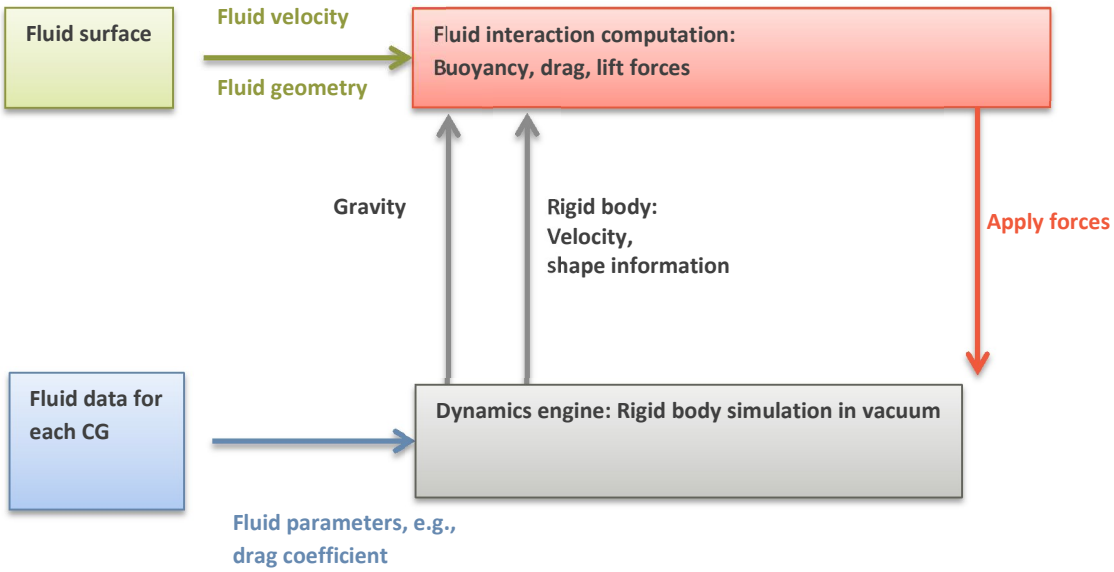

Figure 27. Fluid dynamics simulation: Information flow in Vortex Software fluid dynamics simulation

Contrary to the fluid dynamics effects listed above, the added mass effect is not modeled as a generalised force but incorporated into the rigid body mass matrix. More details on the methods used to simulate the mentioned hydrodynamics effects are provided in the following sections.

6.1 Buoyancy

Buoyancy produces an upward force opposing the weight force of the immersed rigid body. The buoyancy force vector $\mathbf{b}$, acts through the center of the submerged volume and can be computed using the following constitutive equation:

$$\mathbf{b} = -\rho V \mathbf{g} \tag{50}$$

Here, ρ denotes the fluid's density, V is the volume of the immersed portion of the rigid body, and the vector $\mathbf{g}$ corresponds to the gravity force. This scenario is shown in Figure **28**.

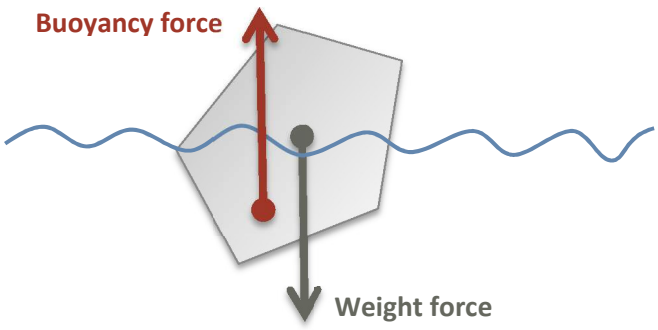

Figure 28. Buoyancy force applied at center of immersed volume opposing weight force.

If the rigid body is above the liquid surface, no forces are generated since the submerged volume drops to zero; if it is fully submerged, the entire volume of its geometry is used to compute the buoyancy force. Otherwise, if the rigid body overlaps the fluid surface, the exact submerged volume and its center is computed. Note that the volume computation produces best results when using a general mesh for the representation of the rigid body's shape.

6.2 Drag

Drag yields a force opposite to the relative surface velocity of a rigid body inside a fluid. For a given surface element, such as the triangle of a general mesh, the drag force vector $\mathbf{d}$ can be computed using the following constitutive equation:

$$\mathbf{d} = -\frac{1}{2}\rho d A \|\mathbf{v}\|^2 \frac{\mathbf{v}}{\|\mathbf{v}\|} \tag{51}$$

In the above expression, d denotes the drag coefficient, ρ is the fluid's density, $\mathbf{v}$ is the local surface velocity of the rigid body relative to the fluid, and A represents the cross section area of the surface element perpendicular to the relative velocity vector $\mathbf{v}$. This situation is depicted in Figure **29**.

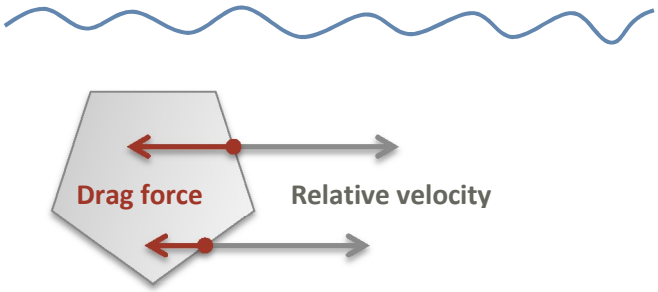

Figure 29. Drag forces applied to a rigid body moving in a fluid with a given relative velocity. The drag force is opposite to the relative velocity at the respective surface patch and is proportional to the surface's cross section area perpendicular to the relative velocity.

If a mesh is used to represent the rigid body's shape in interaction with the fluid, a drag force is computed for every submerged surface triangle, and the resultant forces are integrated. For reasons of numerical accuracy, it is recommended to use a mesh with close to equilateral triangles for best results. Note that both the linear and the angular velocity of the rigid body are accounted for by computing a relative linear velocity for every surface triangle individually.

6.3 Lift

Similar to drag, lift produces a force proportional to the relative velocity of the rigid body to the fluid. As opposed to drag, lift acts perpendicular to the relative velocity at the respective surface patch. Accordingly, the lift force $\mathbf{l}$ can be computed using the following constitutive equation:

$$\mathbf{l} = -\frac{1}{2}\rho l A \|\mathbf{v}\|^2 \ \frac{\mathbf{v}}{\|\mathbf{v}\|} \times \left(\frac{\mathbf{n}}{\|\mathbf{n}\|} \times \frac{\mathbf{v}}{\|\mathbf{v}\|}\right) \tag{52}$$

Here, l denotes the lift coefficient, ρ is the fluid's density, $\mathbf{v}$ represents the velocity of the rigid body at the surface patch relative to the fluid, $\mathbf{n}$ defines the normal of the surface patch, and A is the cross section area of the surface patch perpendicular to the relative velocity vector $\mathbf{v}$. This situation is depicted in Figure **30**.

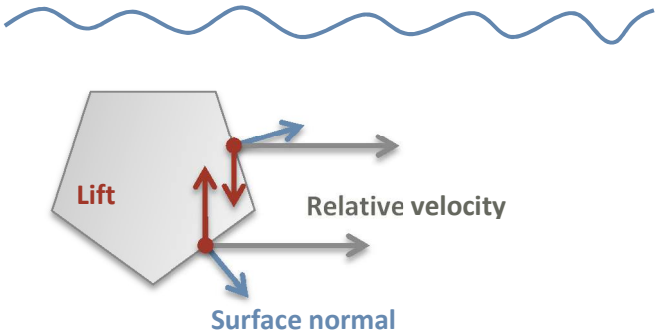

Figure 30. Lift forces applied to surface elements of a rigid body moving in fluid. The lift force is perpendicular to the relative velocity at the respective surface patch.

As can be seen in the expression above, the lift force applied at a given surface patch is computed using a double cross-product between the surface normal and the relative velocity. Consequently, if the surface normal is collinear or perpendicular to the relative velocity, no lift is produced.

Analogously to the computation of drag forces, in order to obtain lift forces with the highest possible physical accuracy, a general mesh with close to equilateral triangles should be used to represent the immersed body.

6.4 Added Mass

Added mass is the inertia added to an immersed rigid body due to the surrounding fluid. Intuitively, if the rigid body is accelerating or decelerating it has to move a portion of the surrounding fluid. As a result, the acceleration of the body is affected.

The added mass effect can be considered as a force which is proportional to the rigid body's acceleration. However, simulating the added mass by applying generalised forces, as done for the other hydrodynamics effects described above, tends to be a source of instability. The instability results from the fact that the applied forces have a feedback effect on the body's acceleration which is used to compute the forces in the first place. Depending on the size of the simulation time step, this approach can lead to the computation of very approximate, excessive forces, which can make the system blow up.

Instead, for improved stability, Vortex models the added mass effect as a modification of the generalised mass matrix of the rigid body (see equation (3) in the *Multibody Dynamics Simulation* section of this guide). This approach is motivated by the work of Weißmann and Pinkall [36], which proposes a stable simulation approach for underwater rigid body dynamics.

For this purpose, we define the 6×6 Kirchhoff tensor $\mathbf{K}_i$ of the ith rigid body as the sum of the generalised mass matrix $\mathbf{M}_i$ and the added mass tensor $\mathbf{K}_{F,i}$, where the subscript F stands for fluid. It is defined as

$$\mathbf{K}_i = \mathbf{M}_i + \mathbf{K}_{F,i} \tag{53}$$

The added mass is considered in the simulation by replacing the generalised mass matrix $\mathbf{M}_i$ of the i'th rigid body in equation (3) (see the *Multibody Dynamics Simulation* section of this guide) by the Kirchhoff tensor $\mathbf{K}_i$. This way, the added mass effect is made part of the multibody dynamics formulation and integrates naturally with the constraint-based modeling approach.

Taking a closer look at a single rigid body and using the definition of the Kirchhoff tensor $\mathbf{K}_i$ in equation (53), we write the updated equations of motion for rigid body i as

$$\mathbf{K}_i \dot{\mathbf{u}}_i = (\mathbf{M}_i + \mathbf{K}_{F,i}) \dot{\mathbf{u}}_i = \mathbf{g}_i \tag{54}$$

where $\mathbf{g}_i$ contains the net generalised forces and gyroscopic terms for the ith rigid body, and $\dot{\mathbf{u}}_i$ denotes the body's generalised acceleration vector, which assembles the 3×1 linear and angular accelerations, $\dot{\mathbf{v}}$ and $\dot{\boldsymbol{\omega}}$, in a single vector:

$$\dot{\mathbf{u}}_i^T = [\dot{\mathbf{v}}^T \quad \dot{\boldsymbol{\omega}}^T] \tag{55}$$

In order to get the desired added mass effect from the surrounding fluid, an appropriate added mass tensor $\mathbf{K}_{F,i}$ must be computed in every simulation step and updated in equation (54). The added mass tensor depends among other things on the submerged volume. When the submerged volume drops to zero as the rigid body leaves the fluid, so does the added mass matrix. Consequently, the added mass effect vanishes as expected. Here, we omit the details of how to compute appropriate added mass tensors and refer you to [36].

To understand why the added mass tensor is suitable for modeling the added mass effect of the fluid, we will take a closer look at its shape. The added mass tensor $\mathbf{K}_{F,i}$ is symmetric and has the following form:

$$\mathbf{K}_{F,i} = \begin{bmatrix} \mathbf{L} & \mathbf{P} \\ \mathbf{P}^T & \mathbf{A} \end{bmatrix} = \begin{bmatrix} l_{11} & l_{12} & l_{13} & p_{11} & p_{12} & p_{13} \\ l_{12} & l_{22} & l_{23} & p_{21} & p_{22} & p_{23} \\ l_{13} & l_{23} & l_{33} & p_{31} & p_{32} & p_{33} \\ p_{11} & p_{21} & p_{31} & a_{11} & a_{12} & a_{13} \\ p_{12} & p_{22} & p_{32} & a_{12} & a_{22} & a_{23} \\ p_{13} & p_{23} & p_{33} & a_{13} & a_{23} & a_{33} \end{bmatrix} \quad (56)$$

Since the added mass tensor is added to the generalised mass matrix, as shown above, it directly affects the inertial forces of the rigid body on the left-hand side of equation (54).

With this observation, and considering equations (54) and (55), it becomes clear that the tensor $\mathbf{K}_{F,i}$ creates coupling effects between different components of the rigid body's generalised acceleration vector $\dot{\mathbf{u}}_i$. Depending on which terms are non-zero in $\mathbf{K}_{F,i}$, specific components of the rigid body's acceleration are affected. As such, the sub-matrix $\mathbf{L}$ couples the linear acceleration terms, the sub-matrix $\mathbf{A}$ couples the angular acceleration terms, and the sub-matrix $\mathbf{P}$ couples the linear with the angular acceleration terms.

As described at the beginning of this section, the added mass of the fluid surrounding the rigid body has a very similar coupling effect on the body's acceleration. Consequently, by choosing the terms in the added mass tensor appropriately, the added mass effect of the fluid can be modeled with the proposed approach. For a detailed discussion, see [36].

7 REFERENCES

[1] CM Labs Simulations, *User Documentation,* Vortex Software, version 6.7, 2016.

[2] T. C. Hudson, M. C. Lin, J. Cohen, S. Gottschalk and D. Manocha, "V-COLLIDE: accelerated collision detection for VRML," in *Proceedings of the second symposium on Virtual reality modeling language*, ACM, 1997.

[3] CM Labs Simulations, *Technical Documentation,* Vortex Software, version 6.7, 2016.

[4] J. B. Marion, Classical Dynamics of Particles and Systems, Academic Press, 2013.

[5] K. Erleben, *Stable, Robust, and Versatile Multibody Dynamics Animation,* Department of Computer Science, University of Copenhagen, 2005.

[6] J. Bender, M. Müller, M. A. Otaduy and M. Teschner, "Position-based Methods for the Simulation of Solid Objects in Computer Graphics," Eurographics STAR, 2013.

[7] J. Baumgarte, "Stabilization of constraints and integrals of motion in dynamical systems," *Computer methods in applied mechanics and engineering,* vol. 1, no. 1, pp. 1-16, 1972.

[8] U. M. Ascher, H. Chin, L. R. Petzold and S. Reich, "Stabilization of constrained mechanical systems with daes and invariant manifolds," *Journal of Structural Mechanics,* vol. 23, no. 2, pp. 135-158, 1995.

[9] M. Anitescu and F. A. Potra, "A time-stepping method for stiff multibody dynamics with contact and friction," *International journal for numerical methods in engineering,* vol. 55, no. 7, pp. 753-784, 2002.

[10] R. Goldenthal, D. Harmon, R. Fattal, M. Bercovier and E. Grinspun, "Efficient Simulation of Inextensible Cloth," *ACM Transactions on Graphics (TOG),* vol. 26, no. 3, p. 49, 2007.

[11] I. Alduán and M. A. Otaduy, "SPH Granular Flow with Friction and Cohesion," in *Proceedings of the 2011 ACM SIGGRAPH/Eurographics symposium on computer animation*, 2011.

[12] M. Silcowitz, S. Niebe and K. Erleben, "A nonsmooth nonlinear conjugate gradient method for interactive contact force problems," *The Visual Computer,* vol. 26, no. 6-8, pp. 893-901, 2010.

[13] R. Weinstein, J. Teran and R. Fedkiw, "Dynamic simulation of articulated rigid bodies with contact and collision," *IEEE Transactions on Visualization and Computer Graphics,* vol. 3, no. 12, pp. 365-

374, 2006.

[14] S. Niebe and K. Erleben, Numerical Methods for Linear Complementarity Problems in Physics-Based Animation, Morgan & Claypool Publishers, 2015.

[15] D. K. Pai, "Strands: Interactive simulation of thin solids using cosserat models," *Computer Graphics Forum,* vol. 21, no. 3, pp. 347-352, 2002.

[16] CM Labs Simulations, *Vortex Software Solution 6.6 Verification and Validation,* Montreal: CM Labs Simulations, 2016.

[17] H. Naunheimer, B. Bertsche, J. Ryborz, W. Novak and P. Fietkau, Automotive Transmissions - Fundamentals, Selection, Design and Application, Springer Science & Business Media, 2010.

[18] J. Y. Wong, Theory of Ground Vehicles, 3rd ed., New York: John Wiley, 2001.

[19] A. Azimi, D. Holz, J. Kövecses, J. Angeles and M. Teichmann, "A Multibody Dynamics Framework for Simulation of Rovers on Soft Terrain," *Journal of Computational and Nonlinear Dynamics,* vol. 10, no. 3, 2015.

[20] H. B. Pacejka and I. J. M. Besselink, "Magic formula tyre model with transient properties," *Vehicle system dynamics,* vol. 27, no. S1, pp. 234-249, 1997.

[21] E. Fiala, "Seitenkräfte am rollenden Luftreifen," *VDI Zeitschrift,* vol. 96, p. 973, 1954.

[22] R. W. Allen, T. J. Rosenthal and H. T. Szostak, Analytical Modeling of Driver Response in Crash Avoidance Maneuvering: An Interactive Tire Model for Driver/Vehicle Simulation, U.S. Department of Transportation, National Highway Traffic Safety Administration, 1988.

[23] M. G. Bekker, Introduction to Terrain-Vehicle Systems, The University of Michigan Press, 1969.

[24] J. Y. Wong, Terramechanics and off-road vehicle engineering: Terrain behaviour, off-road vehicle performance and design, Amsterdam, the Netherlands: Butterworth-Heinemann, 2009.

[25] D. Holz, T. Beer and T. Kuhlen, "Soil Deformation Models for Real-time Simulation: A Hybrid Approach," in *Workshop on Virtual Reality Interaction and Physical Simulation (VRIPHYS)*, Eurographics Association, 2009.

[26] D. Holz, *Deformable Terrain for Real-Time Simulation,* Aachen, Germany: Diploma Thesis, RWTH Aachen University, Virtual Reality Group and Computer Graphics & Multimedia Group, 2010.

[27] D. Holz, A. Azimi, M. Teichmann and S. Mercier, "Real-time simulation of mining and earthmoving operations: A level set-based model for tool-induced terrain deformations," in *Proceedings of the 30th ISARC*, Montreal, 2013.

[28] D. Holz, "Parallel Particles (P2): A Parallel Position Based Approach for Fast and Stable Simulation of Granular Materials," in *11th Workshop on Virtual Reality Interaction and Physical Simulation (VRIPHYS)*, J. Bender, C. Duriez, F. Jaillet and G. Zachmann, Eds., Eurographics Association, 2014.

[29] A. R. Reece, "The Fundamental Equation of Earth-Moving Mechanics," in *Proceedings of the Institution of Mechanical Engineers*, 1964.

[30] E. McKyes, Soil Cutting and Tillage, Elsevier, 1985.

[31] J. Shen and R. L. Kushwaha, Soil-machine interactions: a finite element perspective, Marcel Dekker Inc., 1998.

[32] B. M. Das, Advanced Soil Mechanics, CRC Press, 2013.

[33] M. F. O'Sullivan and E. A. G. Robertson, "Critical state parameters from intact samples of two agricultural topsoils," *Soil and Tillage Research,* vol. 39, no. 3, pp. 161-173, 1996.

[34] D. Holz, A. Azimi and M. Teichmann, "Virtual Reality Simulation of Vehicles and Tools Interacting with Deformable Terrain," in *Movimento Italiano Modellazione e Simulazione (MIMOS 2012)*, 2012.

[35] D. Holz, A. Azimi and M. Teichmann, "Mobility Prediction of Rovers on Soft Terrain: Effects of Wheel- and Tool-induced Terrain Deformations," in *Adaptive Mobile Robotics, Proceedings of the 15th International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines (CLAWAR 2012)*, 2012.

[36] S. Weißmann and U. Pinkall, "Underwater rigid body dynamics," *ACM Transactions on Graphics (TOG) 31.4,* p. 104, 2012.