PRESSURE FIELD CONTACT

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Ryan Blake Elandt

December 2022

PRESSURE FIELD CONTACT

Ryan Blake Elandt, Ph.D.

Cornell University 2022

Pressure field contact (PFC) generalizes the flat Winkler elastic foundation model to contact situations with arbitrary, non-convex, coarsely meshed solid geometries. PFC calculates the deformed contact surface, the traction distribution on that surface, and, consequently, the net wrench (the net force and moment) between the objects. This model is useful in situations where deformation in the contact region is qualitatively important, but need not be accurate.

PFC is a model of, and not a convergent approximation of, continuum mechanics. In PFC, each of the two contacting objects is assigned an immutable internal virtual 'pressure' field. This continuous scalar field is zero on an undeformed object's outer boundaries and increases with depth. This pressure field is chosen to indirectly mimic physical properties such as (possibly spatially varying) elastic constants or a compliant layer's thickness. In regions where two rigid objects nominally overlap, the PFC contact surface is defined as the set of points where the two pressure fields are equal. This compliant unilateral contact model can add deformation details to otherwise rigid-object simulations.

BIOGRAPHICAL SKETCH

Ryan Elandt was born in 1991 in Antioch, California. He attended the University of California at Berkeley (UCB) starting in 2010. In 2014, he received two bachelors degrees from UCB; one in mechanical engineering, and another in materials science and engineering. Later that year, he enrolled in the Ph.D program in Mechanical Engineering Department at Cornell University.

To my parents

Rachel and Ronald Elandt

ACKNOWLEDGEMENTS

I would like to thank my advisor, Andy Ruina, for his support and patience. I would also like to thank my committee members, Steve Marschner and Herbert Hui, for their helpful feedback on this dissertation. Finally, I would like to acknowledge Marcia Sawyer for her years of advice, guidance, and support.

I would like to thank members of the Drake Dynamics group at the Toyota Research Institute. In particular, I'd like to recognize Michael Sherman. The contact model in this dissertation was created based on the requirements in a document he wrote. He also encouraged me to develop the principled compliant-compliant extension of this model. I appreciate the countless hours Evan Drumwright spent helping me better communicate the method. Finally, I would like to thank Sean Curtis for his discussions and insights into computational geometry.

I'm sincerely appreciative of the fellowship support I received while at Cornell. I was supported by the Department of Defense, Office of Naval Research, National Defense Science and Engineering Graduate (NDSEG) Fellowship, 32 CFR 168a.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION TO DISSERTATION

The topic of this dissertation is pressure field contact, or PFC for short. Depending on the context, PFC refers to one of two distinct things. In a continuum mechanics context, PFC is a generalized elastic foundation model that uses pressure fields to model compliant regions. In a numerical simulation context, PFC is an implementation of this continuum model.

I began to develop PFC during a summer internship at the Toyota Research Institute. My project as an intern was to design a contact model for the Drake physics engine to better simulate conforming contacts involving complex geometry, especially the triangular meshes common in engineering. The motivation was to be able to efficiently use polyhedral contact geometry (e.g., boxes and triangular meshes) in simulation-based control design.

Simulation-based controller design relies on optimization, and is most efficient when simulations are free of implementation-caused discontinuities. Traditionally, when using polyhedral contact geometry, implementation-caused discontinuities were unavoidable due to sudden changes in contact point locations.

I first started creating a contact model by designing a simplified continuum elasticity model that possesses certain properties of physical systems (e.g., produces continuous forces, conserves elastic potential energy). In hindsight, the continuum model I created is a generalized elastic foundation model. I call this model Pressure field contact (PFC) because it uses pressure fields to model compliant regions. I used this continuum model to create a PFC implementation for numerical simulation.

The PFC implementation calculates a contact surface that separates two contacting polyhedral objects. It selects point contacts across this surface in a way that changes continuously with state, allowing it to avoid the sudden forces changes typical of polyhedral contacts. This implementation produces contact forces that are continuous (with respect to state) to *machine precision* and differentiable almost everywhere. Arbitrary polyhedral geometry is allowed; meshes can be complex, non-convex, or both.

This dissertation is arranged as follows. In Chapter 2, I describe where PFC fits into the world of contact models. In Chapter 3, I describe the limitations of ad-hoc contact models. In Chapter 4, I arrive at the continuum PFC model by generalizing the elastic foundation model. I describe how to add friction and normal damping in this chapter. In Chapter 5, I describe an implementation of the continuum PFC model. In Chapter 6, I describe how to add tangential compliance to PFC for the purposes of providing an alternative way to add friction to the discrete model. Finally, in Chapter 7, I discuss two limitations of PFC.

PFC was first introduced in [12], a 2019 conference paper. This dissertation improves on this initial explanation in that it: is explicit about what features of the PFC model are important and why; arrives at the pressure field model by generalizing the elastic foundation model in a principled way; contains an improved and expanded handling of damping; contains a detailed pragmatic approach (complete with examples) of how to create pressure fields; contains four dynamics demonstrations that use PFC; and describes an additional way to implement Coulomb friction.

An open source C++ implementation of PFC is available in Drake (`https://drake.mit.edu`). As of 2022-08-24, this PFC implementation is the result of

255 GitHub pull requests, and contains 31,500 lines of code spread over 110 files.

CHAPTER 2

INTRODUCTION TO PFC

This introduction has two purposes. The first is to explain the niche that PFC occupies in the world of contact models. The second is to highlight that engineering simulators select contact models (e.g., equality constraints, point-contact, the finite element method, PFC, etc.) on a per-contact basis. Being able to select from a collection of contact model allows a simulation to be accurate at contacts that require detailed modeling, and fast at contacts that don't.

Consider a rigid-object simulation of planets that orbit the sun. These objects apply gravitational forces to each other and do not touch. For this situation, no contact model is required, but this is atypical. Most applications require some sort of contact model for objects that e.g., touch sporadically during collisions, or touch permanently at joints.

When objects touch they deform, forming a shared contact surface. The net contact force is the result of the traction (i.e., distributed forces) across this contact surface. So, at its core, modeling contact is about modeling the deformation of elastic solids. In engineering, how accurately this deformation needs to be modeled depends on the context. For some applications, the deformation of elastic solids needs to be solved for in detail, while for others it can be neglected entirely.

2.0.1 Elastic-objects and hardware design

Robots tend to be made up of parts that are connected at joints [20]. Parts tend to be designed to be lightweight, yet stiff enough to be modeled as rigid (e.g., [2]). The

goal is for parts to be 'functionally' rigid in that they only undergo minimal elastic deformation under load, and do not vibrate at frequencies likely to be excited by the controller. Deformations are sometimes calculated rigorously, to ensure that parts that should be functionally rigid actually are, or to make an existing part as light as possible [45]. During the design process, detailed elastic modeling is typically done with commercial software packages (e.g., COMSOL) that employ implementations of the finite element method (FEM). For a robot that exists in physicality, experimental modal analysis (see [19]) can be done to identify specific hardware issues, as was described for the bipedal robot LOLA in [2].

2.0.2 Rigid-object simulators for video games

For control purposes, the natural way to model robots made up of functionally rigid parts is as a collection of rigid-objects. Early control design attempts used rigid-object physics engines designed for video games [47].

Video game physical engines typically handle both joint and non-joint contacts in a similar way (e.g., as explained in [44] for ODE). In a velocity-stepping context, constraints are applied at the velocity level (e.g., [6]). Velocity-level linearization errors result in the gradual accumulation of position error. Baumgarte stabilization is sometimes used to prevent the accumulated position error from getting too large (e.g., [6]).

Irrespective of implementation details, these approaches apply forces/impulses to limit kinematic violations that occur when (1) rigid-objects overlap and (2) joint-attached objects move in non-motion directions. For example, consider the kinematic violation that occurs when a falling box intersects the floor. To prevent

this violation from becoming too large, a video game physics engine might apply impulses at each corner of the box. If the box and the floor separate, constraint forces are no longer required.

Constraints that limit kinematic violations between joint-attached objects are always active. For example, consider a revolute joint. This type of joint allows one rotational DoF between two objects. Modeling this type of joint requires five kinematic constraints, one for each non-motion direction.

The problem with this constraint-based approach to joint modeling is that it is too inaccurate for many engineering applications. It allows too much deformation in non-motion directions. In the 2012 MoJoCo paper [47], Todorov wrote:

> ODE as well as other game-oriented engines (such as NVIDIA PhysX and Bullet Physics) represent the system state in over-complete Cartesian coordinates and enforce joint constraints numerically. This is sensible when simulating a large number of mostly disconnected bodies with few joint constraints, however it becomes both inaccurate and inefficient when simulating elaborate multi-joint systems such as humanoids.

To simulate stiff machined joints common in engineering, a different approach to modeling joint-contact is needed.

2.0.3 Rigid-object simulators with analytic joint modeling

MuJoCo and other engineering physics engines use two contact models; one for non-joint contacts, and another for joint-contacts. For non-joint contacts, they

use the kinematic constraint-based approach described above. For joint-contacts, they use rigid and workless kinematic equality constraints, as explained in [17]. Kinematic equality constraints eliminate all joint deformation in non-motion directions. The constraint forces for 'kinematic trees' are algebraically eliminated by rigid-object dynamics algorithms. These structurally recursive algorithms are computationally efficient. The improved accuracy of modeling joints with equality constraints enables users to simulate scenarios such as complex articulated rigid-object systems (e.g., a humanoid robot) as they interact with objects in their environment with the accuracy needed for some engineering applications [47]. Using a new contact model at poorly-modeled joint contacts allows rigid-object simulators to gain functionality and accuracy.

2.0.4 Optimization and discontinuities

Given that engineering rigid-object simulators can accurately model a robot and its joints, the next thing to do is use the simulation to design a controller. Manual control design in the presence of contact and friction is tedious and time-consuming [28]. Ideally one would like to use some type of optimizer to design a controller automatically. Concerning automatic control design, in the 2012 MoJoCo paper [47], Todorov wrote:

> The essence of control optimization is to automatically construct many candidate controllers, evaluate their performance in simulation, and use the data to construct better controllers.

Typically, optimization approaches used to design controllers rely on derivative information [46]. Some approaches perform a gradient update on the con-

troller with gradient information obtained using a finite difference approximation (e.g., [46]). Other approaches (e.g., [20, 35, 5, 39]), obtain gradient information using automatic differentiation (not to be confused with symbolic differentiation). Other 'policy gradient' approaches use a sampling-base approach to perform a gradient-update on the policy (i.e., controller) without needing to explicitly differentiate the dynamics (e.g., [42, 41]). No matter the approach, the gradient tells the optimizer the direction to tune each control parameter to improve performance.

In order for this tuning process to be efficient, the objective function should be well-behaved. And in order for an objective function to be well-behaved, it needs to use a rigid-object simulator that is also well-behaved. And in order for a simulator to be well-behaved, it needs to not suffer from implementation-caused discontinuities [9].

Implementation-caused discontinuities are discontinuities caused by a simulator's adaptive or sampling-based components, and are not part of a scenario's underlying dynamics (e.g., rigid collisions). For example, if a simulator uses an adaptive time-step, small changes in initial conditions can cause sudden changes in a simulation's time-step sequence. To avoid this issue, optimization applications typically use a fixed time step. Another type of implementation-caused discontinuity affects point-force contact models. As explained in detail in §2.0.5, contact points selected by point-force contact models can change suddenly for polyhedral geometry.

2.0.5 Polyhedral geometry and discontinuous forces

Point-force contact models are used in most video game physics engines (e.g., Unreal, Unity) as well as in many robotics simulators (e.g., ODE, Bullet, MuJoCo). These contact models discourage interpenetration by applying point-forces at isolated points. Point-force models use features determined from the overlapping region of the two contacting objects' undeformed geometries to calculate the location(s), direction(s), and magnitude(s) of point-forces. In their simplest form, point-force models determine a single point-force per object pair using each object's most deeply interpenetrated point (i.e., the point on one object furthest inside the other). For example, for contacts between two spheres, a point-force model might apply forces to the two most deeply interpenetrated points, with a magnitude proportional to the distance between these points.

When at least one object is non-convex, applying just a single separative point-force at the single pair of most deeply interpenetrated points can yield a contact wrench that is far wrong. For example, in order to be at equilibrium, a table needs to be supported on multiple legs. If, in a point-force model, at each instant in time, support is limited to just one leg, then a dynamic simulation will predict chattering, as the contact point changes from being at one leg to another at successive time steps. While this bouncing can be fixed if anticipated (e.g., by making a table as a union of 5 objects, 4 legs and a top, with each of the legs having one contact with the floor), it is problematic if unaddressed.

For commonly occurring primitives, like cubes, special rules can be used to determine multiple simultaneous contact points, such as at the corners of a box. Supporting a box with point-forces at each corner is sensible if the box is resting on a half-space, or if two stacked boxes are aligned. But in other situations other

ad hoc rules are needed. For example, if the stacked boxes mentioned above are relatively rotated about the vertical axis, points of force application need to be chosen with another heuristic (e.g., at edge-edge intersections). With enough carefully selected heuristics and treatment of special cases, the unstable chattering of the boxes can be solved. For polyhedral objects in general, when handled carefully, simulations that use point-force contact models can be stable, visually plausible, and exhibit minimal artifacts [14, 25].

Due to the selection of discrete contact points, point-force models typically don't yield forces that are continuous with configuration. For example, for the misaligned boxes mentioned above, as relative-pose progresses, a discontinuous change in point-force location, as selected by a given heuristic, will cause a discontinuity in the net contact wrench. Continuous point contact forces are a challenge for polyhedral geometry in general.

Force discontinuities can be avoided if contacts are only allowed to occur between pairs of special simple convex 'primitive' shapes. For example, for contacts between two spheres, or a sphere and a capsule (a cylinder whose ends are half-spheres), applying just a single point-force to separate the most deeply interpenetrated points gives continuous forces. However, for contacts between a capsule and a half-space, this approach does not give continuous forces. The most deeply interpenetrated point can change suddenly (from near one end of a capsule to the other). This discontinuity can be fixed by applying a second point-force (whose magnitude varies appropriately with depth) at the most deeply penetrated point on the least deeply penetrated spherical end-cap.

Because continuous forces are so important to optimization, the robot models used in MuJoCo for machine learning of controllers (e.g., [26]), are constructed

entirely out of capsules and spheres (Yuval Tassa – personal communication). However, if an existing robot has already been described using polyhedral geometry, then approximating it differently, as unions of capsules and spheres is arbitrary and possibly arduous.

2.0.6 Compliant contact functionality for rigid simulators

As described above, modern engineering physics engines use some kind of point-force contact model at non-joint contacts. For contact geometries constructed from spheres and capsules, this contact model is an effective choice for controller design. It is fast and produces continuous forces. However, the capsules and sphere approach to representing complex geometries is not sufficiently physical for certain contacts. This is especially true for contacts that involve complex non-convex objects, have large contact areas, or both. Contacts like this occur in engineering during e.g., grasping tasks [10], or when designing knee replacements [18]. To add compliant contact functionality to engineering simulators, attempts have been made to create fast contact models that produce continuous forces for polyhedral geometry.

2.0.7 The elastic foundation model

Ideally, one would like contact results that are consistent with continuum elasticity. But, solving continuum mechanics equations with FEM and other high-fidelity techniques is computationally expensive, so people look for ways to approximate it [43]. For some contacts, such as the knee replacements of interest in [18], the natural way to approximate elasticity is with the elastic foundation model.

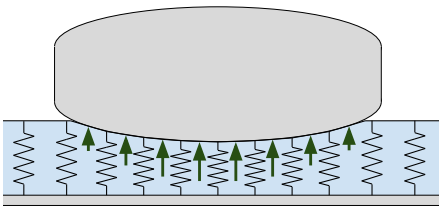

Figure 2.1: **The elastic foundation model.** Contacts between a rigid object and a planar layer (blue). In the elastic foundation model, traction is calculated locally (it only depends on the penetration at that point) and is proportional to penetration distance. For the standard small-angle approximations, vertical and normal tractions are equivalent.

The elastic foundation (EF) or Winkler model estimates the normal traction distribution and deformation of a compliant planar layer, such as a horizontal surface, during contact with a penetrating rigid object [29] – see Fig. 2.1. In the EF model, the layer is a continuum of independent vertical springs that deform vertically. The EF model is well defined in situations where the relative contact-surface slopes are small (i.e., the opposing surface normals are close to anti-parallel). This model can be an accurate approximation of linear elasticity if the elastic layer's thickness is small compared to the other characteristic lengths (i.e., the size of the shared contact surface and the characteristic radii of the opposing surfaces).

The typical elastic foundation model implementation used in dynamics simulations uses discrete springs at fixed locations (e.g., [43]). This approach is computationally problematic because fine discretizations are often required, even for simple contacts. For example, consider the contact between a $10m \times 10m$ elastic layer and a rigid $0.1m$ cube. At any given instant, the bottom of the cube can only touch $1/10,000^{th}$ of the layer's surface. If a minimum of 10 springs need to touch the bottom of the box for good simulated behavior, the layer must contain at least 100,000 springs. As this example highlights, EF implementations that use fixed-location springs tend to require fine discretizations and therefore have slow run-times. Discrete springs can also introduce force discontinuities. For example,

the number of springs supporting the above mentioned box will vary as the box slides across the layer.

An EF model implementation doesn't need to use discrete springs. An implementation could first calculate the polyhedral contact surface exactly from the overlapping polyhedral geometry. Then, it could integrate the traction analytically over this contact surface to obtain the net contact wrench. Although conceptually straightforward, this approach is not easy to implement in practice. As mentioned in [24], analytic integration becomes difficult in the presence of friction. Difficulties associated with analytic integration can be avoided if one approximates the wrench by sampling tractions at discrete points across the contact surface. But sampling points across a contact surface can be problematic, as it can introduce force discontinuities.

As described earlier in §2.0.5, when sampling points across polyhedral geometry, it becomes challenging to pick points in a way that doesn't change suddenly with changes in topology. To see how sudden contact point movement affects an EF model, consider the following scenario. A box is tilted so that one corner is slightly lower than the others. The box's tilt is held fixed as the box is pressed into a planar layer. As the penetration of the box increases, the shape of the contact surface changes. When one corner of the box is submerged, the contact surface associated with the bottom of the box will have three sides. When two corners are submerged, it will have four sides. When three corners are submerged, it will have five sides. Finally, when all four corners are submerged, it will go back to having four sides.

Sudden changes in topology cause sudden changes in contact point locations. In the presence of friction, these sudden contact point changes can cause discontinuous forces. Since getting continuous forces is so important to optimization-based

controller design, people look for non-EF ways of modeling contacts for polyhedral geometry. It's important to emphasize that force discontinuities associated with discrete-spring EF models are implementation-caused. The continuum EF model is well-defined and well-behaved for smooth objects with nearly parallel outer surfaces.

2.0.8 Geometric contact approaches

A family of ad-hoc approaches I call 'geometric contact models' is designed to produce continuous (with respect to state) forces for polyhedral contact geometry. These models which include e.g., [1, 4, 15, 21, 16, 24, 34], calculate contact forces from the geometry of the overlap region (defined as the region enclosed by the undeformed outer-boundaries of two intersecting objects). Geometric features of this overlap region, such as volume, change continuously. These continuously varying features are used to calculate continuous forces.

Calculating forces from geometry allows these approaches to produce forces with some physically realistic features. For example, most of these approaches apply a force that is proportional to the overlap region's volume. Because the overlap volume is a continuously varying quantity, so is the force magnitude. One limitation of these methods is that their use is restricted to contacts between convex objects. Another limitation is that they fail to capture sometimes important features of conforming elastic contacts (e.g., accurate scrub torques) because they don't calculate a deformed contact surface or traction distribution. A detailed discussion of geometric contact models is included in Chapter 3.

CHAPTER 3

AD-HOC CONTACT MODELS AND THEIR LIMITATIONS

The following is a detailed description of some of the shortcomings of existing contact model implementations, some of which the PFC model overcomes.

3.1 The role of contact models in simulation

Rigid-object simulators treat objects as rigid for the purposes of kinematics and dynamics. *For kinematics purposes,* the configuration (geometry) of the system is specified by the pose (position and orientation) of all the nominally-rigid parts [27]. Objects can rotate and translate relative to each other, due to the movement of joints, but individual objects cannot change shape. *For dynamics purposes,* these simulators assume rigid-object kinematics when calculating gravitational forces, linear and angular momentum, energy, and the rates of change of these quantities [27]. Momentum and energy calculations are based only on the rigid-object rotation, CoM location, mass, and moment of inertia.

In rigid-object simulations, forces come from gravity, springs, dampers, motors, contact at joints between robot parts (e.g., at revolute or prismatic joints), and from other contacts between objects (e.g., a foot and the ground).

Contact at joints is assumed to obey rigid-object kinematics [17]. That is to say, the play and compliance of joints are assumed to be small enough to be neglectable. Such ideal rigid contact is described with kinematic *equality* constraints on the relative generalized positions, velocities, and accelerations of joint-connected parts. These equality constraints are enforced with always-active con-

straint forces. Standard methods either eliminate (i.e., using minimal coordinate methods) or quickly find (i.e., using differential-algebraic-equation methods) these forces (see [17]).

On the other hand, **non-joint contacts** occur between disjoint objects such as a robot's foot and the ground. These contacts can apply separative normal forces only. Such contacts are therefore said to be unilateral [17]. Few rigid-object simulators employ a truly rigid contact model for unilateral contacts. For example, MuJoCo cannot simulate truly rigid contacts, but it can simulate contacts that are "phenomenologically hard" [47]. Reasons for using compliant unilateral contact models range from computational expediency (as mentioned in [17]) to physical realism. No matter the reason, the major function of compliant contact models is to calculate contact forces from overlap geometry. The most popular type of contact model are those that apply point-forces – previously described in §2.0.5.

In statically indeterminate situations, the details of local compliance, not just the existence of compliance, can have a large effect on the motion and forces. For example, the predicted motions of Newton's cradle are qualitatively different depending on whether a linear or nonlinear contact model is used (e.g., [7]). The analytic formula for the force-penetration relationship for non-conforming contacts between elastic spheres was first described by Hertz [29]. But, formulas don't exist for contacts in general. That being said, contact forces typically increase with increasing contact area and overlap volume. Neglecting these effects, even qualitatively, can cause a simulation to fail to be predictive, as the Newton's cradle example highlights.

In the next section, I discuss a family of approaches that uses the overlapping contact geometry to capture how contact forces typically increase with increasing

contact area and overlap volume.

	Normal force Magnitude	Direction	Location
Gonthier[21]	$\propto V$	eigenvector[2]	intersects centroid
Hasegawa[24]	$\propto V$	vector area	intersects centroid
Luo[34]	appx. $\propto V$	vector area	intersects centroid
Faure[16]	$\propto V$	integrated constant pressure	
Allard[1]	constraint[1]	integrated constant pressure	

Table 3.1: **Geometric contact model approaches to finding the contact force.** A few models are reviewed. Columns are (1) force magnitude, (2) force direction, and (3) location of the application point. In each column, equivalent methods share the same color. Integrated constant pressure spans two columns because it determines direction and location simultaneously.

3.2 Geometric contact models

When simulating polyhedral geometry, "geometric contact" models calculate contact forces from continuously varying geometric quantities in order to capture nonlinear compliance, and alleviate force discontinuities associated with point-force contact. Geometric contact methods use the overlap region (defined in §2.0.8 as the region enclosed by the undeformed outer-boundaries of two intersecting objects) to calculate a normal force: (1) magnitude, (2) direction, and (3) application point (e.g., [1, 4, 15, 21, 16, 24, 34]).

Table 3.1 summarizes how some geometric contact models calculate the normal force magnitude, direction, and application point. **Force magnitude.** Because more nominal overlap should imply more force, many models apply a force proportional to volume [4, 16, 24, 34]. However, in a velocity-stepping context the contact force magnitude is solved for as a constraint[1] [1, 15]. **Force direction.** In [21], a

[1]The constraint force makes the overlap volume at the end of a time-step not exceed a certain value.

normal force direction is determined by an eigenvector (principal direction) of the overlap region[2] Other approaches use an area-weighted normal of one of the two sides of the contact region (both sides yield the same direction) [24, 34]. The area-weighted normal is the same normal calculated with a 'volume gradient' approach (e.g., [1, 4, 15, 16]) because these volume gradient methods apply a uniform pressure to the surfaces of the overlap region. **Location of contact force.** For some geometric contact approaches, the normal force passes through the centroid of the overlap region [21, 24, 34]. Other approaches calculate the net contact wrench by integrating a uniform pressure on the outer boundary [1, 4, 15, 16].

3.2.1 Limitations of point-contact and geometric contact

Geometric contact models address point-contact's issues with continuous forces for polyhedral geometry. But, neither geometric nor point-force contact models calculate a sensible contact surface or traction distribution.

In actuality, contact tractions are applied to the shared surface of objects deformed by mutual compression. The contact surface is a compromise of the local shapes and curvatures of the two objects, with the resulting shape favoring that of the stiffer object. Point-force contact models, by definition, don't calculate a contact surface or traction distribution. The contact surface predicted by geometric contact models, to the extent that one exists, isn't necessarily similar to either object, it is always flat. In addition, the location of this flat contact surface *doesn't* depend on the relative compliance of the two objects. This means that for rigid-compliant collisions, geometric contact will predict a contact surface that

[2]The moment of inertia of the overlap volume has three eigenvectors. This approach selects the eigenvector that is closest to a geometric estimate provided by a commercial software package.

intersects the rigid-object at least half the time.

For point-force and geometric contact models, not being able to calculate a sensible contact surface and traction distribution leads to incorrect predictions of torsional moments. For example, these moments are important when picking up and reorienting a pencil with a thumb and index finger. Some physics engines that use point-force contact models neglect torsional friction entirely. Other physics engines model torsional moments with an *ad hoc* torsional friction coefficient [3]. For example, physics engines such as MuJoCo use a constant torsional friction coefficient. A constant torsional friction coefficient makes the maximum frictional torque proportional to the normal force, for an assumed contact area.

Hardware tests are required to find, or at a minimum verify, the correctness of assumed constant torsional friction coefficients. Such *ad hoc* tuning for torsion is necessary because contact wrenches follow from the traction distribution, which point-force models don't calculate. Geometric contact models require a similar *ad hoc* approach to calculate torsional moments. Because, as described above, the contact surface and traction distribution produced by geometric contact is deficient at best, and missing at worst.

3.3 Conclusion

Contact models that require hardware tuning in order to be predictive typically have limited extrapolation utility. A tuned torsional friction model will be most accurate for the hardware and scenario for which it was calibrated. If a contact model needs to be tuned on hardware, it is not clear how to use such a model to inform the design of future hardware. Similarly, if a contact model needs to

be tuned for a specific scenario, automated control design efforts may encounter one of two problems. The first problem is that the optimizer may synthesize a control strategy that cannot be implemented on real hardware because this strategy uses smaller-than-calibrated-for contact areas, which causes grasped objects to unexpectedly reorient. The second problem is essentially the opposite of the first. If the control strategies that work best on real hardware use larger-than-calibrated-for contact areas, an optimizer may not find these strategies, as it thinks they don't work. Another more fundamental issue with not having access to a contact surface and traction distribution is that it is difficult to predict measurements from say a particular pressure sensor.

In order to avoid the issues stemming from not having a contact surface and traction distribution, robotics applications sometimes use contact models that generalize the elastic foundation model described in §2.0.7. I discuss possible generalizations to the elastic foundation model in the next chapter.

CHAPTER 4

PRESSURE FIELD CONTACT (CONTINUOUS MODEL)

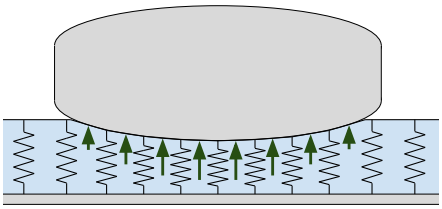

Figure 4.1: **The elastic foundation model.** Contacts between a rigid object and a planar layer (blue). In the elastic foundation model, traction is calculated locally (it only depends on the penetration at that point) and is proportional to penetration distance. For the standard small-angle approximations, vertical and normal tractions are equivalent.

As described in §2.0.7, the elastic foundation EF model applies to small-angle contacts involving a planar layer (see Fig. 4.1). PFC generalizes the elastic foundation model to: (1) arbitrary angles, (2) non-layers (for 3D compliant objects), and (3) compliant-compliant contact.

4.1 Generalizing the planar EF model to arbitrary angles

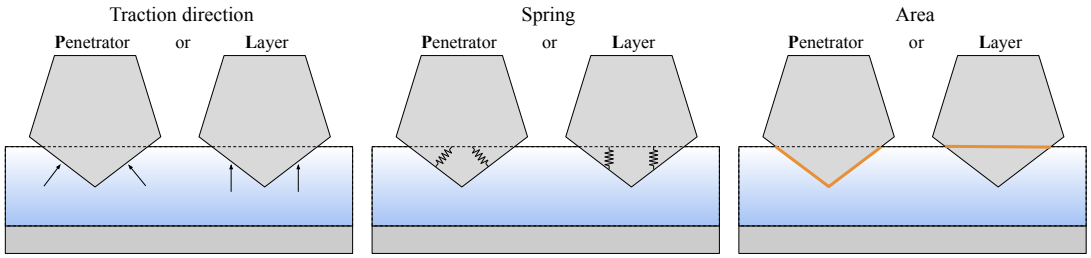

Figure 4.2: **Generalizing the EF model to arbitrary angles.** To calculate traction, an EF model needs a traction direction, a spring stretch, and an area. This figure shows how a generalized EF model can use either the penetrator P or the layer L to calculate the traction direction (left), spring stretch (middle), and area (right). These three choices are independent.

The EF model does not have a unique interpretation for contacts with large

relative angles. To apply an EF model to situations with large relative angles, one has to choose whether to use the penetrator P or the layer L to calculate the traction direction, spring stretch, and area, as shown in Fig. 4.2. These choices are independent, so there are eight different ways to generalize the EF model to arbitrary angles.

4.2 Eliminating problematic generalizations

Traction direction / Spring / Area	Sensible forces §4.2.1	Algebraic equal §4.2.2	Closed-loop test §4.2.3
PLP	✓		✓
LLP	✓		✗
LLL	✓		✗
PPL	✓	PLP	
LPL	✓	LLP	
LPP	✗		
PPP	✗		
PLL	✗		

Table 4.1: **The different ways to generalize the planar uniform stiff layer EF model to arbitrary angles.** P and L denote whether the given feature inherits its properties from the layer L or the penetrator P. See Fig. 4.2.

The goal of §4.2 is to find which of the eight possible generalized EF models is the most sensible. I arrive at the best model by process of elimination. This process is summarized in Table 4.1. Later in §4.3, I develop one of these models into PFC, a well-behaved general-purposes EF model for arbitrary angles and non-planar compliant regions.

The elimination process has three rounds. In §4.2.1, I start with the eight models, and eliminate three of them because they do not produce sensible fores, leaving five models. Then, in §4.2.2, I eliminate two more models because each is algebraically identical to a model that is more physically sensible, leaving three

models. Finally, in §4.2.3, I eliminate two more models because they generate energy in a closed-loop cycle, leaving just one model.

4.2.1 Round 1: Sensible forces in static scenarios

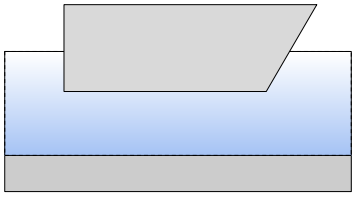

Figure 4.3: **A problematic scenario for some generalized EF models.** An asymmetric indentor penetrates a compliant layer. Three of the eight generalized EF models produce non-sensible forces for this scenario.

A generalized EF model should produce sensible forces, even for large angles. Figure 4.3 shows a scenario for which three of the eight models in Table 4.1 don't do this. The last three models in the table fail. Models LPP and PPP fail because they apply infinite forces to the left side of the boat. These models have an infinite spring stretch and a finite contact area at this surface. The PPP model is the generalized EF model used in [43]. The PLL model fails because it produces a net force to the left. This model applies no force to the vertical left side of the boat, and an upper-left force to the sloped right side. There is a net force to the left. The boat should be in equilibrium. This is the generalized EF model used by [11].

4.2.2 Round 2: Algebraic model equivalence

In §4.2.1, I eliminated three of the eight possible generalized EF models for not producing sensible forces. The upper five models in Table 4.1 remain: PLP, LLP, LLL, PPL, and LPL. In this section, I eliminate two of these remaining models:

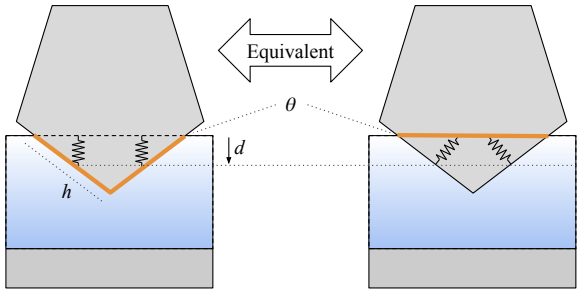

Figure 4.4: **EF model equivalence.** An indentor penetrates a compliant layer. The traction magnitude produced by both cases in dh. **Left:** last two letters LP – the spring stretch is d. The area is the penetrator area h. **Right:** last two letters PL – the spring stretch is the $d/\cos\theta$. The area is $h\cos\theta$.

PPL and LPL. These two models have P as the middle letter, meaning that they calculate the spring stretch based on the deformation of the rigid penetrator (see Fig. 4.2). Calculating spring stretch this way doesn't make physical sense, as it's the layer that deforms, not the penetrator. Not making physical sense is a reason for elimination because it prevents these models from being generalized to a variable stiffness layer. Little is lost by eliminating them. For planar layers, each is algebraically equivalent to a model that calculates spring stretch using the layer (see Fig. 4.4).

4.2.3 Round 3: variable stiffness layer

Five of the eight possible EF generalizations were eliminated in either §4.2.1 and §4.2.2. The upper three models in Table 4.1 remain: PLP, LLP, and LLL. Each of these three models uses the layer to calculate spring stretch. For these methods, I drop the concept of explicit springs, and treat stiffness as a property of the layer. The layer stiffness does not need to be constant. It can be spatially varying, as shown in Fig. 4.5.

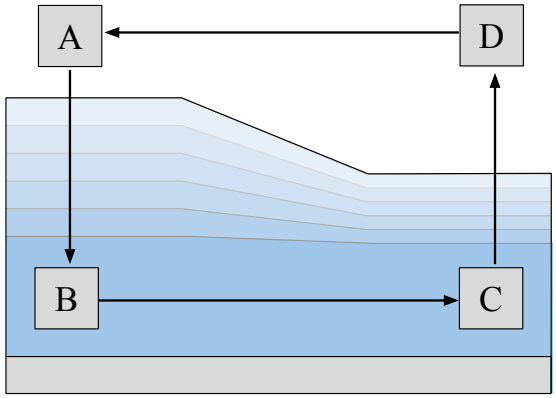

Figure 4.5: **A closed loop cycle (A-B-C-D-A) that causes non-conservative generalized elastic foundation models to generate energy.** Points A and D are outside of a variable-thickness layer. On path A-B, the stiffness of the layer increases linearly, until it reaches a maximum constant value. On path B-C, the block travels sideways through the constant stiffness region. On path C-D, the stiffness of the layer decreases linearly, until it exits the layer. For a method to be conservative, the energy input along A-B needs to be equal to the energy output on C-D.

The closed cycle in Fig. 4.5 causes all but the PLP model to generate energy. All three models do no work as the object moves along paths B-C and path D-A. Energy is lost on path A-B and gained on C-D. The LLP and LLL models apply all tractions upward, and therefore gain less energy on C-D than they consume on A-B. The PLP model applies tractions inward to the object. For this model, the net force is proportional to the stiffness gradient, so the PLP model generates the same energy on C-D that it consumes on A-B. This conservative approach is discussed further in the next section.

4.3 Pressure fields

Only the PLP generalization of the EF model is well-behaved for variable stiffness layers. In the PLP model, tractions are applied inward, as pressures. The traction

magnitude doesn't depend on the penetrator angle (see Fig. 4.4). So, at each point in the layer, there is a traction magnitude that would/will be experienced if a penetrator reaches that point, independent of the orientation of the penetrating surface. In other words, each point in the layer has a unique pressure. It follows that for this model, the layer can be thought to possess an immutable body-fixed *pressure field*. I call this approach pressure field contact or PFC because compliance in this model can be thought of in terms of a pressure field.

By reasoning physically about pressure fields, PFC can be extended to arbitrary rigid-compliant contacts, as well as compliant-compliant contacts. The purpose of a pressure field $p_0(\boldsymbol{r})$ is to model the traction magnitude one would expect at point $\boldsymbol{r}$ based on the extent of material compression. Subject to the general restrictions that $p_0 = 0$ on the outer surface and that p_0 generally increases with depth, the pressure field $p_0(\boldsymbol{r})$ can be chosen to suit a range of situations. To preclude objects from passing into each other, pressure can rise dramatically with depth. For a variable thickness layer, one can allow the isobars to conform to the thickness variations (as seen in Fig. 4.5). For a uniform thickness layer, with horizontally varying material properties, one can scale each vertical slice's pressure field by the local intrinsic material stiffness. The pressure field approach is flexible. For example, pressure fields can even be constructed to give contact forces that agree with elasticity for non-compliant contacts (see Appendix B).

For non-layers, the thought process behind creating pressure fields is the same. The pressure field should vary in a way that makes sense for the material and geometry. For example, for a box, one might make a pressure field that increases linearly with distance from the edge. To preclude objects from passing through each other, one could make this pressure field increase nonlinearly. For arbitrary

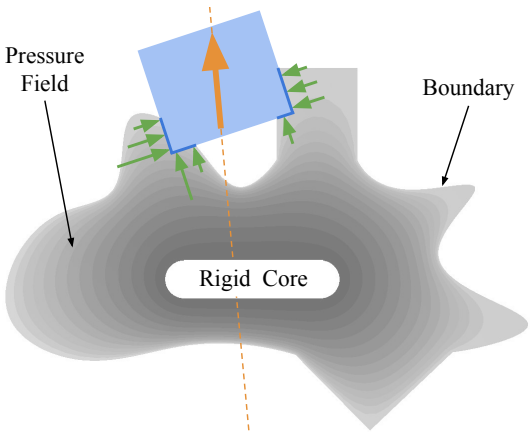

Figure 4.6: **Rigid-compliant contact.** The pressure field for a non-trivial 2D geometry is visualized with gray isocontours. A rigid box (blue) penetrates a partially compliant object comprised of a deformable exterior (gray) and a rigid core (white). Green arrows show the pressure that the compliant object exerts on the rigid blue object. The net force on the blue box, from the compliant object, is shown in orange. The pressure field gradient is normal to the isocontours. This pressure field was generated with Laplace's equation (see text).

objects, generating a suitable pressure field seems challenging at first glance, but it is actually straightforward. As discussed later in §5.1.1, one way to generate smooth fields is with solutions to Poisson's equation, with $p_0 = 0$ on the outer surface and $p_0 > 0$ specified on some interior surface. The pressure field for the object in Fig. 4.6, was generated this way.

4.3.1 Contact between a rigid indenting object and a compliant non-planar object

For arbitrary rigid-compliant (rigid indenter, compliant substrate) contacts as shown in Fig. 4.6, PFC operates the same way as it does for planar contacts; it applies the substrates pressure field pressure to those parts of the indenter's boundary that penetrate the substrate.

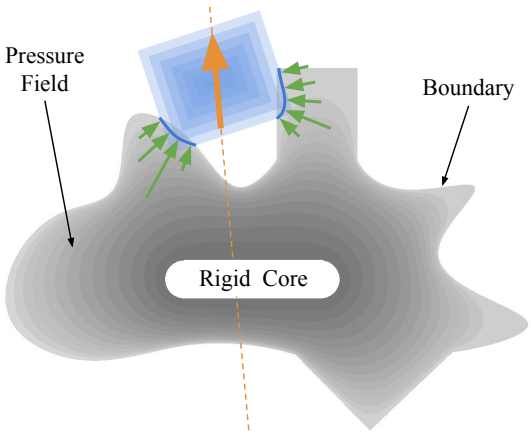

Figure 4.7: **Compliant-compliant contact.** The pressure field for a non-trivial 2D geometry is visualized with gray isocontours. A blue compliant box interpenetrates a partially compliant object comprised of a deformable exterior (gray) and a rigid core (white). Deformed object boundaries touch at the blue line. Green arrows show the tractions the compliant object exerts on the rigid geometry. The net force on the blue box, from the compliant object, is shown in orange.

4.3.2 Compliant-compliant contacts

For compliant-compliant contacts, each deformable object has its own immutable body-fixed pressure field (see Fig. 4.7). On the contact surface, contract tractions are integrated to get the net force and moment. In order to satisfy action/reaction, these contact tractions must be equal and opposite. Therefore, the contact surface is defined as the set of points in the overlap region where the two pressure fields match. That is, the PFC contact surface $\mathcal{S}$ is defined to be the set of all points where the spatially varying pressure fields p_0 of compliant objects A and B are equal (Fig. 4.7). In other words, $\mathcal{S}$ is the set of all points $\boldsymbol{r}$ where $p_{0_A}(\boldsymbol{r}) = p_{0_B}(\boldsymbol{r})$.

Within this approach to compliant-compliant contact, a rigid indenter A is a limiting case of a compliant indenter. It has a pressure field that increases from zero on its outer boundary, to some high value, higher than any p on the interior of the penetrated substrate B, on an interior shell of A, infinitesimally interior to the outer boundary of A. So the equality condition $p_{0_A}(\boldsymbol{r}) = p_{0_B}(\boldsymbol{r})$ is met in the

limit, at the outer boundary of A.

4.4 Calculating the contact wrench

The net contact moment and force (net wrench) *applied by* B, is represented by the net wrench $\boldsymbol{f}$, a 6-component vector (3 moment and 3 force components). The wrench $\boldsymbol{f}$ is the integral sum of traction $\boldsymbol{t}$ contributions over the PFC contact surface, as shown in (4.1),

$$\boldsymbol{f} = \int \begin{bmatrix} \boldsymbol{r} \times \boldsymbol{t} \\ \boldsymbol{t} \end{bmatrix} d\mathcal{S}, \tag{4.1}$$

where $\boldsymbol{r}$ is the position, relative to a fixed origin, of points on the PFC contact surface. In the absence of damping and friction, traction comes from pressure field contributions only as follows:

$$\boldsymbol{t} = p_0 \hat{\boldsymbol{n}}, \tag{4.2}$$

where $\hat{\boldsymbol{n}}$ is the outward normal of the contact surface relative to the object acted on by $\boldsymbol{t}$. When damping (discussed in §4.4.2) and friction (discussed in §4.4.3) are added, the total traction $\boldsymbol{t}$ applied to A is the sum of the normal traction $\boldsymbol{t}_{\hat{\boldsymbol{n}}}$ and a frictional traction $\boldsymbol{t}_f$ as follows:

$$\boldsymbol{t} = \boldsymbol{t}_{\hat{\boldsymbol{n}}} + \ \boldsymbol{t}_f. \tag{4.3}$$

Assuming smooth-enough normal damping and sliding friction models, and continuous velocities (i.e., no rigid-rigid collisions away from the PFC contact), PFC yields a net wrench that is continuous with both state and time (see Appendix C.5).

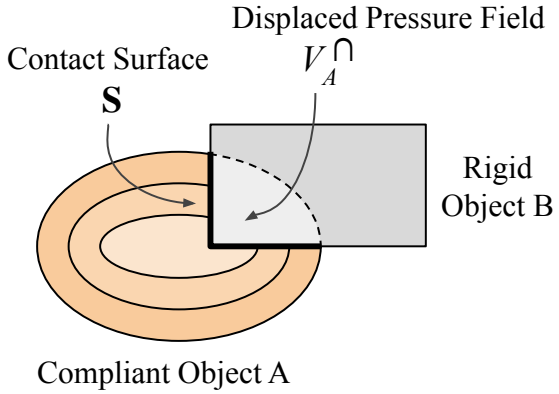

Figure 4.8: **Contact between a compliant ellipse and a rigid rectangle.** The PFC contact surface $\mathcal{S}$ is an interface between objects with A on one side and B is on the other. In the contact scenario pictured above, part of object A's pressure field lies on object B's side of the contact surface. The part of object A's pressure field that lies on B's side of the contact surface is said to be displaced. The displaced volume nominally occupied by A's pressure field is denoted by $V_A^{\cap}$.

4.4.1 Energetically conservative

PFC is energetically conservative. During contact in PFC, a portion of the pressure fields of compliant objects gets displaced (see Fig. 4.8). The displaced volume nominally occupied by A's pressure field is denoted by $V_A^{\cap}$. The potential energy associated with PFC is the energy required to displace pressure fields. As derived in Appendix A.1 for rigid-compliant contacts, the PFC potential energy is the integral of the pressure field over the displaced volume $V_A^{\cap}$ as shown:

$$U_A^{p_0} = \int_{V_A^{\cap}} p_{0_A}(x, y, z)\ dx\ dy\ dz. \tag{4.4}$$

For compliant-compliant contacts, the strain energy is the sum of the strain energy required to displace both pressure fields: $U_A^{p_0} + U_B^{p_0}$.

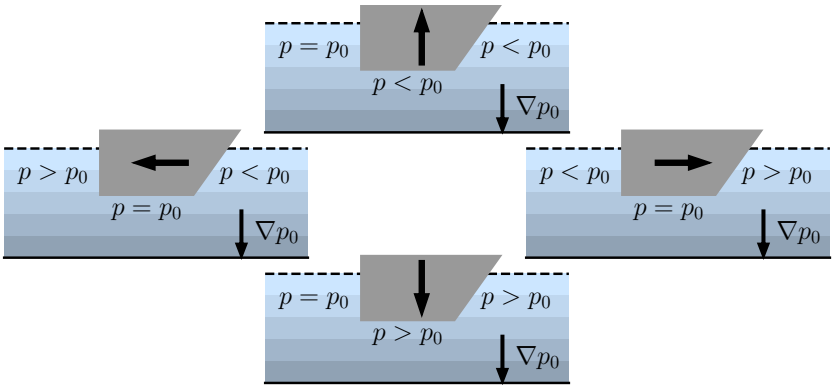

Figure 4.9: **Damping trends in PFC** visualized using a gray boat-shaped indenter that translates in four different directions in a pressure field p_0 (blue). The net pressure p includes contributions from p_0 and a damping term p_χ. That is $p = p_0 + p_\chi$ as long as the sum is positive. The dashed line marks where $p_0 = 0$; darker blues denote higher values of p_0. The pressure on the left, bottom, and right sides of the boat should behave as marked.

4.4.2 Normal dissipation

Damping is needed to inhibit vibrations in simulations, to model rolling resistance, to model dissipation in collisions, and to model hysteresis in gripping and ungripping. At each point on the equal-pressure PFC contact surface $\mathcal{S}$, the pressure depends on the nominal pressure field p_0 and the dissipation pressure p_χ as follows:

$$p = \max(p_0 + p_\chi, 0). \tag{4.5}$$

The max() function precludes negative pressure (i.e., suction) when objects separate and $p_\chi < 0$ would be greater in magnitude than p_0. The boat in Fig. 4.9 shows the expected damping behavior for rigid-compliant contact. A more precise explanation of p_χ follows.

Let Q denote a point on the PFC contact surface $\mathcal{S}$. I use $\boldsymbol{v}_A$ and $\boldsymbol{v}_B$ to denote the velocity of fixed points in the frames of B and A that are instantaneously coincident with Q. I use $\boldsymbol{v}_{B/A}$ to denote the relative velocity at Q of B with

respect to A as follows:

$$\boldsymbol{v}_{B/A} \equiv \boldsymbol{v}_B - \boldsymbol{v}_A. \tag{4.6}$$

This relative velocity concerns the relative velocities of particles on the bodies assuming rigid-object kinematics. It is not related to the velocity of the contact surface itself, which evolves in ways related to the gradient of p_0.

Let $\hat{\boldsymbol{n}}$ denote the outer normal from A at point Q. At each point on the PFC contact surface, the damping term p_χ opposes $\boldsymbol{v}_{\hat{\boldsymbol{n}}}$, the component of velocity normal to $\hat{\boldsymbol{n}}$ as shown:

$$\boldsymbol{v}_{\hat{\boldsymbol{n}}} = (\boldsymbol{v}_{B/A} \cdot \hat{\boldsymbol{n}})\hat{\boldsymbol{n}}. \tag{4.7}$$

The velocity dependent damping term p_χ is greater than zero in regions that are approaching, and less than zero in regions that are separating. Section 5.2.2 describes a practical candidate constitutive law for $p_\chi(v_n, p)$. The net normal traction is the pressure applied in the normal direction as follows:

$$\boldsymbol{t}_{\hat{\boldsymbol{n}}} = p\hat{\boldsymbol{n}}. \tag{4.8}$$

4.4.3 Friction

As explained above, the normal dissipative pressure p_χ resists the relative normal velocity of the two objects. Friction resists the relative tangential velocity of the two objects. Denoted by $\boldsymbol{v}_T$, the relative tangential velocity of B with respect to A, i.e., $\boldsymbol{v}_{B/A}$, projected onto the PFC contact surface $\mathcal{S}$ at a point Q is:

$$\boldsymbol{v}_T \equiv \boldsymbol{v}_{B/A} - \left(\boldsymbol{v}_{B/A} \cdot \hat{\boldsymbol{n}}\right) \hat{\boldsymbol{n}}, \tag{4.9}$$

where $\hat{\boldsymbol{n}}_A$ denotes the PFC contact surface normal at Q that points out of A. The friction traction $\boldsymbol{t}_f$ acts on A, and oppositely on B, in the tangent plane. In the case of isotropic friction, $\boldsymbol{t}_f$ acts in the direction opposite of the tangential sliding $\boldsymbol{v}_T$. The particular choice of friction model (e.g., the relation between normal traction, sliding velocity, and friction traction) is not fundamental to the PFC model, and can be chosen freely. Section 5.2.3 describes a regularized Coulomb friction model, where friction force is a continuous function of velocity. Chapter 6 describes how to add tangential compliance to PFC in order to support Coulomb friction without the need for regularization.

4.5 Numerical experiments

I investigated the pressure distribution and accuracy of forces produced by PFC for contacts between a compliant layer and a rigid indentor. Of interest in this study were scenarios where the layer is thin (i.e., thinner than the contact surface is wide), and the indentor penetration was small compared to the layer thickness. For these scenarios, I modeled the layer using a *linear* pressure field with a modulus $E^* = \frac{E(1-\nu)}{(1+\nu)(1-2\nu)}$. The process of creating nonlinear pressure fields to model contact forces between an elastic half-space and a rigid indentor is discussed in Appendix B.

The purpose of this section is to communicate the following ideas: 1. PFC is quantitatively close to FEM, for contacts between a 'thin' compliant layer and a rigid sphere or cylinder indentor. 2. sphere and cylinder indentors are sensible models for real-world objects (marble and lip balm container), 3. the PFC contact surface shares trends with the FEM contact surface, and in this regard is superior to point-contact (which has no contact surface), and 4. one can use geometry infor-

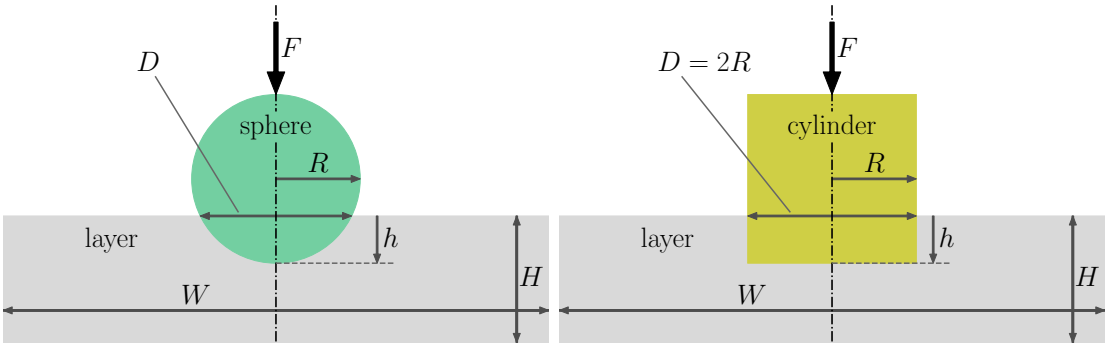

Figure 4.10: **Schematic of contacts between two two rigid objects and a compliant layer.** Both scenarios are rotationally symmetric.

mation to estimate the quantitative accuracy of PFC for a given contact scenario. It's my hope that this section provides practical guidance to those who use PFC in their simulations.

The contact scenarios considered in the study are shown in Fig. 4.10. For these scenarios, the radius of the rigid object is R, the thickness of the compliant layer is H, and the maximum penetration of the rigid object into the layer is h. The contact surface diameter as calculated by PFC is D (see Fig. 4.10). For the cylinder, the PFC contact surface diameter is twice the radius, so $D = 2R$. With regard to comparing different models, there are different regimes depending on the relative sizes of R, H, h, and D. For the small penetration case (i.e., $h \ll R$) of interest in this study, the PFC contact surface diameter is $D \approx \sqrt{8Rh}$.

4.5.1 FEM model

Small strain FEM calculations were performed in `ANSYS` assuming that $\nu = 0.3$. The bottom of the compliant layer was fixed. The layer's vertical wall (away from the axis of rotation) was fixed horizontally and free to move vertically. The assumed frictionless contact surface was calculated with the 'normal Lagrange'

contact formulation. Both scenarios were meshed with quadratic elements as described below.

The geometry used in the FEM model of the sphere-layer contact scenario from Fig. 4.10 is as follows: layer width $W = 40$, radius $R = 1600$, and layer thickness $H = 1$. The penetration h is a prescribed boundary condition and is adjustable, but is much smaller than both H and R. The PFC contact surface diameter D is a function of the tunable penetration h. The geometry was meshed so that the element size was 0.015 at the contact surface. The maximum allowed element size was 0.2.

The geometry used in the FEM model of the cylinder-layer contact scenario from Fig. 4.10 is as follows: layer width $W = 60$, and radius $R = 1$. The layer height H is adjustable, and the penetration h is fixed to be $h = H/100$. The geometry described above was meshed so as to resolve the high stresses at the outer edge of the contact surface. An element size of 0.001 was used to mesh the outer 0.02 of the contact surface, as well as a distance 0.02 beyond it. An element size of 0.01 was used for the remainder of the contact surface. The maximum allowed element size was 0.4.

In order to quantify the accuracy of PFC for a range of H/D values, the above-mentioned FEM models were simulated for a range of parameters. As described above, each FEM model used a fixed-size indentor. For the cylinder, the contact surface diameter D was constant, and H was varied to produce a range of H/D values. For the sphere, the PFC contact surface diameter D was a function of the penetration h. The layer thickness H was held constant, and the penetration h varied to produce a range of H/D values. These FEM parameter sweeps were used to create Fig. 4.11, discussed in the next subsection.

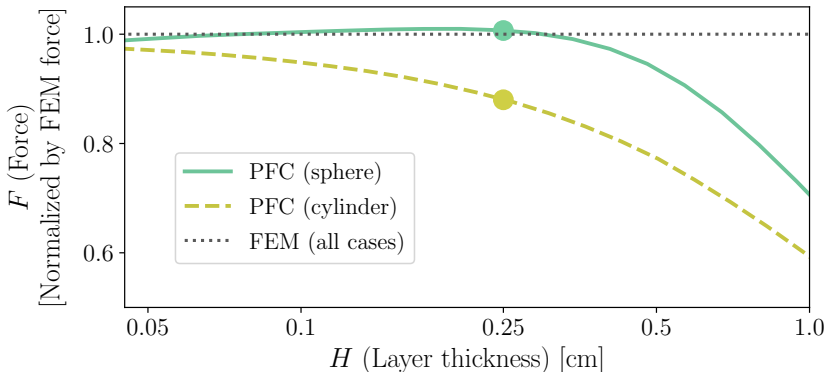

Figure 4.11: **The force accuracy of PFC for the two contact scenarios in Fig. 4.10.** The accuracy of PFC-calculated forces relative to FEM produced by pressing (1) a rigid sphere or (2) a cylinder into a compliant layer assuming a $D = 1$cm contact diameter. The compliant layer was modeled as a linear pressure field with a modulus $E^* = \frac{E(1-\nu)}{(1+\nu)(1-2\nu)}$, see [18]. The contact surface pressure distributions for the marked points at $H = 0.25$cm are shown in panel (b) of Fig. 4.13.

4.5.2 Force accuracy of PFC

The accuracy of PFC-calculated normal contact forces depends on: (1) indentor geometry, and (2) the ratio of the layer thickness H to the contact diameter D. Figure 4.11 shows the accuracy of PFC-calculated contact forces for sphere and cylinder indentors. In this figure, H and D are given units for the purposes of reader clarity, even though this dimensionless relationship depends only on the H/D ratio.

The results in Fig. 4.11 demonstrate that PFC is more accurate when the layer is thin. The accuracy of PFC approaches 100% as the layer thickness approaches zero. The thickest layer considered, $H = 1$cm, has the same dimension as the $D = 1$cm contact surface. For this thickness, the PFC calculated normal force is at least 60% accurate for the cylinder and 70% accurate for the sphere. Irrespective of layer thicknesses, PFC is more accurate for the sphere than the cylinder. I examine these trends and extrapolate the results of this study to other contacts

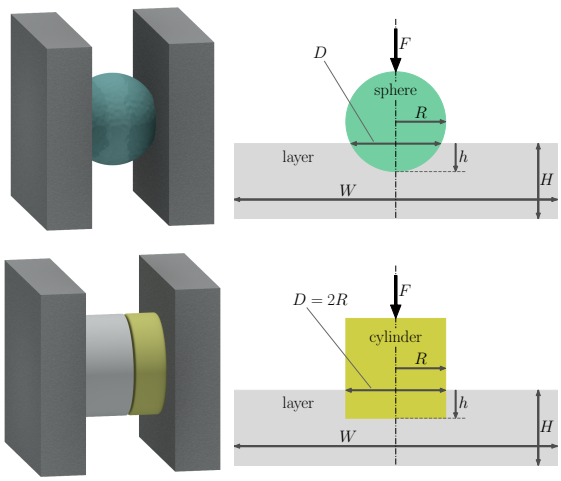

Figure 4.12: **A contact model for two contact scenarios. Panel (a).** Two contact scenarios where a pair of foam grippers hold a teal marble (top) and a white and yellow lip balm container (bottom). The back of each foam pad is attached to a rigid substrate (not pictured). **Panel (b).** Modeling the scenarios in panel (a). The foam grippers are modeled as a compliant layer. The marble is modeled as a rigid sphere. The lip balm container is modeled as a rigid cylinder.

later, in §4.5.5.

4.5.3 Real-world contacts

The contact geometry described in Fig. 4.10 resembles the geometry of some real-world contacts. As shown in Fig. 4.12, a sphere penetrating a soft layer might be a sensible model for a pair of foam grippers holding a marble when the penetration h, pad thickness H, and contact diameter D are related as follows: $h \ll H \ll D$. A cylinder penetrating a soft layer might be a sensible model for a pair of foam grippers pinching the top and bottom of a lip balm container, again when the penetration h, pad thickness H, and contact diameter D are related by: $h \ll H \ll D$.

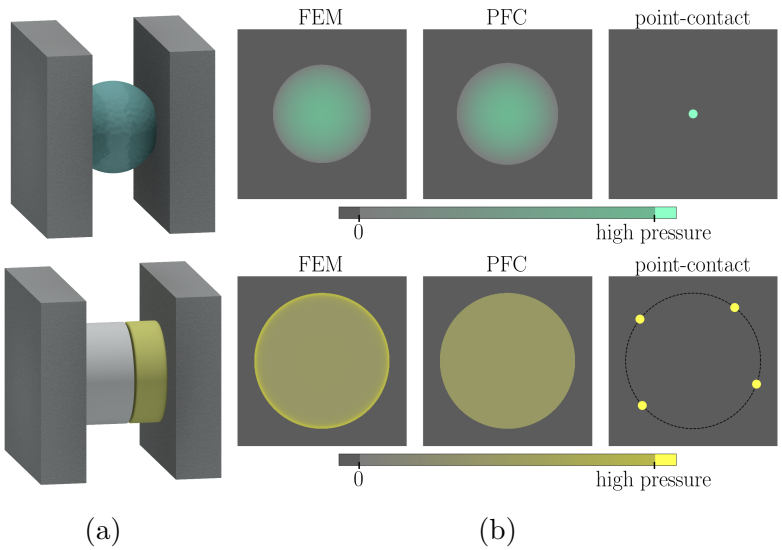

Figure 4.13: **A pressure distribution for two contact scenarios. Panel (a)** shows the contact scenarios involving a teal marble (top) and a yellow lip balm container (bottom) from Fig. 4.12. **Panel (b)** shows the pressure distribution on the interface between the object and the right pad for two scenarios when $H/D = 1/4$ and $h \ll H$; the object and left pad are not shown. This figure looks directly into the right pad. For the sphere (top), the PFC-predicted contact surface is slightly larger than predicted by FEM; the pressure distributions are qualitatively similar. A point-contact model concentrates all the pressure at a single point and captures little about the pressure distribution. For the cylinder (bottom), the PFC predicted contact surface has the correct size, but misses the stress concentrations near the edges. A point-contact model might concentrate all the pressure at points on the edges of the cylinder, and therefore capture little about the pressure distribution.

4.5.4 Contact surface pressure

Figure 4.13 shows the contact surface pressure distributions produced by FEM, PFC, and point-contact for the two contact scenarios in this study. The visualized contact surface pressure distributions were produced using the ratio $H/D = 1/4$. For the case considered in Fig. 4.11, the contact surface diameter $D = 1$cm, so the $H/D = 1/4$ ratio is satisfied when $H = 0.25$cm. This $H = 0.25$cm point for each object in Fig. 4.11 is marked with a circle.

4.5.5 Extrapolating the case study

Conceptually speaking, the pressure distribution for real contacts is the sum of the contributions from: (1) the compression of material below the contact surface and (2) the shear stresses from stretching of material around the contact surface. For a fixed contact geometry and penetration distance, decreasing the layer thickness increases the first term and has little effect on the second term. PFC models just compressive effects, and therefore tends to be more accurate for thinner layers, assuming contact geometry and penetration distance are fixed. Figure 4.13 shows that the FEM-calculated pressure distribution for the cylinder has a ring of high pressure on the outer edge caused by shear stresses from the outer edge stretching the elastic layer. PFC neglects shear-effects, and is therefore more accurate for the sphere where shear-effects are smaller. The ability for PFC to model sphere indentors more accurately than cylinder indentors is demonstrated in Fig. 4.11. The sphere indentor is more accurate than the cylinder indentor for all layer thicknesses considered. These observations can be extrapolated to contacts is general.

The accuracy of PFC depends on size, shape, and geometry. **Size.** The accuracy of PFC depends on the ratio of contact diameter to layer thickness. For example, the accuracy of PFC for the following scenarios is the same: (1) a 2cm diameter cylinder on a 1cm layer and (2) a 1cm diameter cylinder on a 0.5cm layer. **Shape.** If the penetrating object was an ellipsoid instead of a sphere, the accuracy of this contact would be similar to that of a sphere with the same effective radius. **Geometry.** PFC is more accurate for the sphere than the cylinder because the sphere's contact surface has gently sloped edges, while the cylinder has vertical walls. Most contacts possess characteristics that lie somewhere between these two shapes and are likely to have accuracy that lies in the middle as well.

CHAPTER 5

PRESSURE FIELD CONTACT (IMPLEMENTATION)

A PFC implementation contains a variety of essential components. To use PFC one needs: (1) a way to create pressure fields for arbitrary compliant objects; (2) to represent rigid objects and pressure fields for computation; (3) to find the contact surface for the above mentioned geometry; (4) a way to calculate friction and normal dissipation at points on the contact surface; and (5) to integrate traction across the surface to obtain the contact wrench. Three of these components (representing geometry, finding the contact surface, and integrating traction) are inter-related. The choice of geometry affects the shape of the contact surface, which in turn affects how traction is integrated. I discuss these three components first, as they depend on the same computational geometry choices. The remaining components (creating pressure fields, the pointwise calculation of friction, and normal dissipation) are chosen freely and are discussed second.

5.0.1 Part 1: representation of geometry

I represent fully rigid objects with triangular meshes on the outer boundary and compliant objects with tetrahedral meshes on the interior. For example, a compliant sphere might be approximated by an icosahedron (20 faces) using 20 tetrahedrons (tets) that share a common vertex at the center.

I use pressure fields that are linear over each tet. As described in the next sections, this design choice simplifies both the calculation of the contact surface and the integration of traction. For each tet, an arbitrary linear p_0 field is represented by (stored as) the value of p_0 at each vertex. Online, I calculate p_0 at some point

Q in a tet or on its boundary by linearly interpolating the vertex values of the tetrahedron. Neighboring tets share three vertices and vertex values, so linear interpolation in each tet guarantees that p_0 is continuous across tet faces.

Mesh sizing considerations. PFC represents surfaces with tri meshes and volumes with tet meshes. FEM implementations often take the same approach (e.g., see [8]). For rigid objects, FEM and PFC share similar mesh sizing considerations. In particular, meshes need to approximate the shape of the rigid geometry [40]. For example, a rigid sphere might be approximated by an icosahedron (20 triangles).

On the other hand, for compliant objects, the mesh sizing considerations of FEM and PFC are different. For PFC, the contact surface is determined from the overlapping *undeformed* objects. So compliant PFC meshes only need to approximate the nominal shape of compliant objects. Again, a compliant sphere might be approximated using 20 tetrahedrons (tets) that share a common vertex at the center. In contrast, for FEM, the contact surface is the shared interface between *deformed* objects. So compliant FEM meshes need to approximate what objects look like in their deformed state.

The need to only represent the undeformed shape of objects allows PFC to use meshes that are drastically more coarse. In fact, PFC needs just a *single* tet to model a complaint layer. Objects can contact the layer anywhere, as long as the tet is wide enough for objects to not fall off the sides. In contrast, for FEM, the number of tets needed to model a layer can be large enough to be problematic. Contact applications that use FEM typically analyze just a single contact and use scenario-specific information to keep the mesh size to a minimum. For example, the location of contact is typically known ahead of time, so the mesh is made to

be more dense near the contact region, and less dense further away. Commercial FEM implementations, as well as packages such as e.g., [30] attempt to make this process user-friendly. PFC is fast because it is able to use very coarse meshes.

5.0.2 Part 2: finding the contact surface

In PFC, I defined something called 'the contact surface'. The contact surface is the shared surface on which I integrate tractions to get the net interaction force and moment. This surface is determined from the rigid exterior of rigid objects, and the pressure fields of compliant objects.

Rigid-compliant contacts. For rigid-compliant contacts, the contact surface is the portion of the rigid-exterior that lies within the compliant volume. PFC represents rigid-exteriors with triangular (tri) meshes and compliant volumes with tetrahedral (tet) meshes. The contact surface is therefore the portion of the tri-mesh that lies within the tet-mesh. This tri-mesh/tet-mesh intersection is the union of intersecting tri-tet pairs.

The intersection of a tri-tet pair is found with a computational geometry operation called clipping [13]. Conceptually, tri-tet clipping begins with a piece of paper in the shape of the triangle. Then, the parts of the paper triangle that lie outside of the faces of the tet are clipped (i.e., removed with straight-line cuts). The paper remaining (if any) is the tri-tet intersection.

The clipping operation described above is expensive, and is unnecessary if the triangle and tet are far away from each other. To avoid unnecessary clipping operations, I want to only clip tri-tet pairs that are likely to intersect. The process of finding likely intersecting pairs is called 'broad-phase collision detection' [36]. For

the purposes of broad-phase collision detection, I build a bounding volume hierarchy BVH[1] for each mesh (tri-mesh or tet-mesh). Broad-phase collision detection between two BVHs is computationally efficient.

Compliant-compliant contacts. For compliant-compliant contacts, the contact surface is the set of all points where two overlapping pressure field values are equal. PFC represents compliant volumes with tetrahedral meshes and uses a linear pressure fields for each tet. The contact surface is therefore the set of all points where the overlapping linear pressure field values of tet-mesh A and tet-mesh B are equal. This equal pressure field surface is made up of contributions from individual tet-tet pairs.

For tet-tet pairs, the portion (if any) of $\mathcal{S}$ that lies within the intersection of two tets is a convex polygon that lies on a plane (see Appendix C.1). The flat shape of each piece is the result of using linear pressure fields. It's important to point out that although the piece of the contact surface associated with each tet-tet pair is a convex polygon, the full contact surface is, in general, neither flat nor convex. The contact surface needs to completely separate the two contacting objects, and to do this it may need to be non-flat (e.g., for a torus resting on a cone), non-convex (e.g., for a torus resting on a plane), or even disjoint (e.g., for a table resting on a plane).

Finding the polygonal piece for each tet-tet pair is a three step process: (1) find the plane it lies on, (2) clip the plane by the faces of tet A, (3) clip the plane by the faces of tet B. These three steps are computationally expensive and are unnecessary if two tets are far away from each other. As with tri-tet pairs, I only want to perform these operations for pairs that are likely to overlap. I determine

[1] A BVH is a tree-like data structure that recursively groups nearby objects.

likely overlapping pairs, using the same approach I used for tri-tet pairs. I create one BVH for each tet-mesh for use in broad-phase collision detection.

Rigid-compliant as a limiting case of compliant-compliant. For computational purposes I handle the rigid-compliant and compliant-compliant contacts separately. However, the rigid-compliant case can be thought of as a limiting case of compliant-compliant contact. As the stiffness of a compliant object approaches infinity, the level surfaces of constant pressure collapse onto the object's outer boundary. The pressure at the boundary of arbitrarily stiff compliant volume can take on any finite value, so it becomes a rigid boundary.

5.0.3 Part 3: integrating traction

The contact surface is made up of polygons connected at their edges. The traction on each polygon makes a contribution to the net force and torque. Polygons are flat, so each polygon has a constant-at-each-instant surface normal. For frictionless static scenarios, the pressure across each polygon is linear as a result of using linear pressure fields. Within the context of friction-less static scenarios, the integrand (terms inside the integral sign) for the force calculation is linear. The integrand $p_0\hat{\boldsymbol{n}}$ is linear because p_0 is linear and $\hat{\boldsymbol{n}}$ is a constant. Similarly, the integrand for the torque calculation is quadratic. The integrand $\boldsymbol{r} \times p_0\hat{\boldsymbol{n}}$ is quadratic because $\boldsymbol{r}$ is linear, p_0 is linear, and $\hat{\boldsymbol{n}}$ is a constant. It follows that for friction-less static scenarios, the wrench (i.e., force and torque) calculation contains only first and second order terms.

In the presence of friction and normal damping, the wrench calculation involves terms with an order greater than two. In a force-acceleration context, friction is

typically a function of slip velocity that contains either a (1) sigmoid-type function (e.g., tanh), or (2) a piecewise polynomial. When either of the above items are present, analytic integrals over each polygonal piece of the contact surface are likely to be tedious at best. Analytic integration is further complicated by normal damping. Damping increases the contact surface pressure when objects collide. The faster the objects move toward each other, the higher the pressure gets. When the objects separate, the pressure decreases. The faster the objects separate, the lower the pressure gets, *but* pressure cannot be negative, so it saturates at zero at a certain velocity (see (4.5)). Saturation caused by normal damping makes integration non-smooth, further complicating analytic integration. To avoid difficulties associated with analytic integration, I want to approximate the wrench with quadrature.

As explained in depth in [32], quadrature rules approximate integrals as the weighted sum of the integrand evaluated at particular *quadrature points*. Quadrature rules can be applied to simple domains including line segments, triangles, and quadrilaterals. Quadrature rules are said to have an order (e.g., second-order). Quadrature rules are exact for polynomial integrands up to that order. Using quadrature to approximate the wrench contribution for each polygonal piece of the contact surface requires one to: (1) select a quadrature order, and (2) break the polygon into simple shapes[2].

Selecting a quadrature order. For friction-less static scenarios, a second order quadrature rule will calculate the wrench exactly. The integrand contains only first and second order terms. In the presence of normal damping and friction,

[2]Although I was unaware of it at the time, quadrature rules can be created for non-convex polygons as explained in [37]. So, breaking the polygon into simple shapes is not required for functionality. But as far as I'm aware, breaking the polygon into simple shapes is still required, if one wants the approximated integral to be a continuous function of state.

all quadrature orders will be approximate. The integrand is not a polynomial. But even so, a second order quadrature rule is good-enough. The details of friction and normal damping are not known to any reasonable accuracy. So, approximating them to high accuracy is unwarranted.

Tessellating a polygon. To calculate the wrench over a polygon, the polygon first needs to be broken into simple shapes (I used triangles). For each triangle, the contact wrench is calculated with quadrature. The wrench contributions from all the triangles are added together to give the wrench for the polygon. A polygon can be tessellated into triangles in more than one way. Although multiple tessellations are possible, not all tessellations are equally desirable.

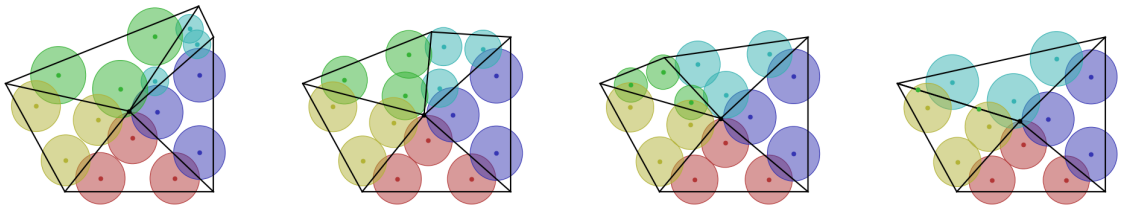

Figure 5.1: **Continuous quadrature over an evolving polygon.** From left to right, a polygon edge grows and vanishes, as does its corresponding triangle. A piece of the polygonal PFC contact surface loses one of its five sides as state evolves in time (left to right). The polygon is divided into triangles that share a vertex at the intersection polygon's centroid. Dots represent the position of quadrature points. The large circles around the dots represent the weight of quadrature points. The weight and position of quadrature points move continuously even as an edge disappears.

As explained in §2, I want to select contact points in a way that changes continuously with topology to get continuous forces for optimization. In PFC, the location of the contact points (i.e., quadrature points), depends on the location of the triangles. The triangularization needs to change continuously to make integration change continuously. In other words, in order to produce continuous forces, the tessellation needs to change continuously, even when the shape and number of

sides of the polygons on the contact surface changes. Our method accomplishes this without remembering past discretizations by tessellating each intersection polygon into triangles, where each triangle has a vertex at the polygon's centroid, and shares one of its edges (see Fig. 5.1). This ensures that only zero-area triangles are added/removed as polygon topology changes. Thus, there is no discontinuity in any surface integrals associated with the creation or deletion of polygon edges. The processes of computing and tessellating the PFC contact surface is covered in Algorithm 1.

Clipping degeneracy. For tet-tet pairs, there is a non-zero probability clipping degeneracy that one needs to handle. This degeneracy occurs because of how the contact surface boundary is created. I will explain why this issue doesn't occur for tri-tet pairs, but does occur for tet-tet pairs.

For tri-tet pairs, the contact surface boundary intersects the zero pressure face of the tet-mesh. In other words, each edge of the boundary is the result of a triangle being clipped by a uniform pressure $p_0 = 0$ tet face. Each edge on the boundary is clipped exactly once, and no degeneracy occurs.

For tet-tet pairs, the contact surface boundary intersects the zero pressure face of both tet-mesh A *and* tet-mesh B. Each edge of the boundary is the result of a plane being clipped by a uniform pressure $p_0 = 0$ face of both tets. Each edge on the boundary is clipped exactly *twice*, this causes the degeneracy. In the presence of numerical error, this duplicate clip can cause one edge to become two.

If not handled, this duplicate clip degeneracy can result in forces that are not continuous to machine precision. My solution to this tet-tet degeneracy is this: if tet A and tet B both have a uniform pressure $p_0 = 0$ face, only clip by the uniform

pressure $p_0 = 0$ face of tet A.

5.1 Parameterizing p_0

To reproduce a linear Winkler elastic foundation, one can use $p_0 = Me$, where M is the relevant modulus, and extent e is a normalized measure of spring compression. In other words, $e = 0$ when the spring is fully stretched, and $e = 1$ when the spring is fully compressed. Extent e increases linearly from 0 to 1 based on the extent of spring compression.

$$p_0 = \underbrace{M}_{\text{scale}} \underbrace{e}_{\text{extent}}. \tag{5.1}$$

For arbitrary objects, I think of extent $e(x, y, z)$ as a normalized measure of pressure field compression. In other words, $e = 0$ for points on the boundary of a compliant object, and $e = 1$ for points on rigid surfaces. Extent e increases from 0 to 1 based on the extent of deformation required to penetrate to that point.

Next, I will describe an intuitive way to create well-behaved e fields for non-convex polyhedral objects.

5.1.1 Creating extent fields

An intuitive way to create well-behaved extent fields involves thinking of extent as a temperature distribution. The steady-state temperature distribution in a conducting solid with internal heat generation (e.g., from radioactive decay) is

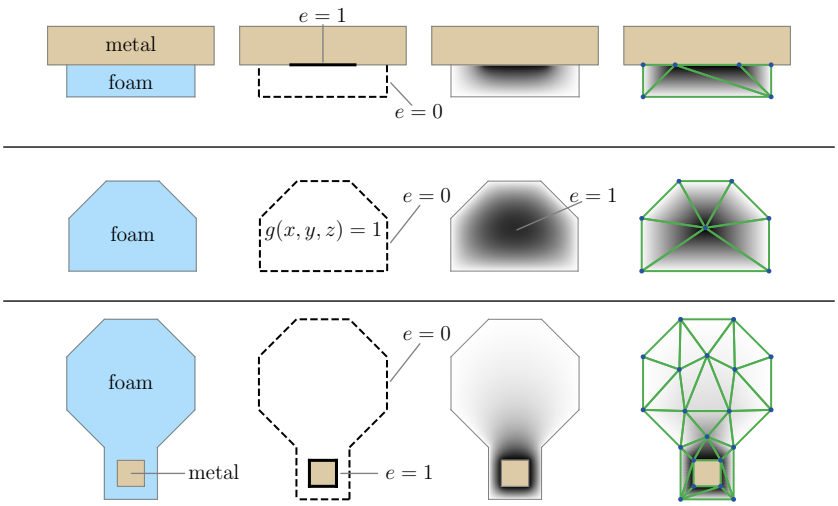

Figure 5.2: **Creating extent fields.** Three examples of how to create an extent field e as described in S5.1.1 with Poisson's equation. The **first column** shows three compound objects: a foam gripper with a metal back (top), a solid foam shape (middle), and a foam paddle with a metal core (bottom). The **second column** shows the boundary conditions, and source term g used when solving Poisson's equation. The **third column** shows the solution to Poisson's equation for the aforementioned boundary conditions. The solution to Poisson's equation for the middle object was normalized so that the solution lies between 0 and 1. The **fourth column** shows each solution approximated with a coarse linear triangular mesh. These coarse approximations are similar to the finer solution in column three.

given by Poisson's equation:

$$\nabla^2 e = \frac{\partial^2 e}{\partial x^2} + \frac{\partial^2 e}{\partial y^2} + \frac{\partial^2 e}{\partial z^2} = \underbrace{g(x, y, z)}_{\text{'source' term}} \tag{5.2}$$

Using this equation to create an appropriate extent field requires setting boundary conditions and, if necessary, selecting a source term. The special case of Poisson's equation where the source term is zero is Laplace's equation.

The first step to creating extent fields with Poisson's equation is to set boundary conditions. The extent e is zero (low temperature) on the compliant volume's outer boundary. The extent e is one (high temperature) at any boundaries that touch the rigid core. All other boundaries are insulated. They use a zero flux boundary condition that says the heat flux through them is zero.

The second step to creating extent fields with Poisson's equation is to determine if a source term is required. A source term is not needed if an object has a rigid core. An object that touches a rigid core (high temperature) will form a temperature gradient as the heat flows to the outer boundary (low temperature). In the absence of a high temperature boundary condition, a heat source is needed to generate a temperature gradient.

I used the Poisson's equation approach described above to determine extent fields for three specific objects in Fig. 5.2. This figure shows how coarse piecewise linear fields can reasonably approximate their finer counterparts.

5.1.2 The pass-through problem and nonlinear p_0

Robotic grippers typically have a soft gripping surface that is attached to a stiff metal backing. The natural way to model such a part with PFC is to treat the soft gripping surface with a linear pressure field, and the stiff metal backing as a rigid core. This approach may work well, assuming that the contact force it provides is sufficiently large to prevent 'pass-through'. In other words, is the force large enough to prevent gripped rigid objects from *passing through* the compliant gripper and penetrating the gripper's rigid backing? This is the pass-through problem.

The pass-through problem can be viewed as either: (1) a theoretical mathematical problem, or (2) a numerical simulation problem. The solution to this problem depends on one's perspective. A solution for each perspective follows.

Theoretical mathematical problem. In order to prevent rigid parts from interpenetrating, the work required to interpenetrate must be ∞. For this to

happen, the pressure or extent field needs to go to infinity, and it needs to do so sufficiently quickly. For pointed indentors, a pressure field that approaches ∞ isn't enough to prevent interpenetration. The needed divergence of pressure to make the work infinite depends on the sharpness of the indentor. Regardless, the idea of crafting pressure fields that prevent penetration by requiring infinite work to displace is purely theoretical.

Crafting pressure fields to prevent pass-through for specific indentor shapes is neither justifiable from material properties, nor is it applicable to numerical simulation. Time-steps have a finite size. So, even if a pressure field went to infinity at an object's boundary, this might not be sufficient to prevent interpretation, as the boundary could be "stepped" over. If the infinite pressure boundary was stepped on exactly, this would cause problems with floating point arithmetic.

Numerical simulation problem. As far as the desired behavior of real robots is concerned, pass-through is a non-issue. In PFC, objects are modeled as rigid not because they are truly rigid, but rather because the relevant contact compliance is assumed to exist elsewhere. For example, consider the robot claw made up of a soft gripping surface attached to a stiff metal backing discussed earlier. The metal backing of such a claw is likely to be modeled as rigid, because it is: (1) much stiffer than the gripping surface, and (2) not expected to participate in contact. If one picks up a metal bearing (modeled as a rigid-sphere), with the above mentioned claw, and pass-through occurs, this means the simulation predicts metal-on-metal contact. If a simulation predicts pass-through (i.e., metal-on-metal contact), this may mean that the compliant object's stiffness is poorly modeled, or that there is an issue with the scenario being simulated (e.g., simulated robot grip force exceeds real actuator limits).

If a simulation predicts pass-through this may mean that the gripper's stiffness is nonlinear. Real materials tend to become stiffer the more they are compressed. Stiffness that increases with compression is especially pronounced for soft gripping surfaces, particularly those made of foam, rubber, or biological materials. I model how stiffness increases with compression by making the modulus a function of compression as shown below:

$$p_0 = \underbrace{M(e)}_{\substack{\text{nonlinear} \\ \text{scale}}} \underbrace{e}_{\text{extent}}. \tag{5.3}$$

To retain the computational advantages of piecewise-linear pressure fields mentioned in §5.0.1, I mesh objects so that they have layers and calculate p_0 at each vertex using an expression for M such as:

$$M = \frac{M_0}{1 - ke}, \text{ where } 0 \leq k < 1. \tag{5.4}$$

If the nonlinear stiffness of a material is well-characterized (e.g., with true strain), this relationship should be used instead.

If stiffness is accurately modeled, and a simulation still predicts pass-through, this may point to a problem with the scenario being simulated. For example, this may mean the gripper force is sufficiently large to cause the mechanical failure of the gripper material in the real world. If this occurs, it could mean that the simulated gripper force exceeds real actuator limits. Or it could indicate that a stiffer gripping surface should be selected.

5.2 Traction calculation

This implementation calculates the contact wrench with quadrature. The technique this method uses to calculate quadrature points was described in §5.0.3. In order to calculate the net contact wrench, I now need to calculate the traction at each quadrature point. In the absence of damping and friction, traction comes from pressure field contributions only as follows:

$$\boldsymbol{t} = p_0 \hat{\boldsymbol{n}}. \tag{5.5}$$

In the presence of damping and friction, the expression changes. This section explains the choices I made for damping and friction that are used to create the expression for $\boldsymbol{t}$ in §5.2.4.

5.2.1 Calculating a normal damping constant for compliant objects

For the purposes of calculating normal dissipation in §5.2.2, I calculate a normal damping constant for each compliant object. I use χ_i to denote the normal damping constant for object i. To calculate χ_i I use two things: (1) an assumed maximum expansion rate τ_i (a material property); and (2) a characteristic length L_i (an object property). Subsequent paragraphs explain these quantities in greater detail. The product of these two quantities is a maximum expansion velocity $v_{\hat{\boldsymbol{n}}_i}^{max}$ as shown:

$$v_{\hat{\boldsymbol{n}}_i}^{max} \equiv L_i \tau_i. \tag{5.6}$$

Physically speaking, the maximum expansion velocity is the rate at which an object's surface would expand if contact tractions dropped to zero. The damping

constant χ_i is the inverse of this maximum expansion velocity.

$$\chi_i \equiv \frac{1}{v_{\hat{\boldsymbol{n}}_i}^{max}} = \frac{1}{L_i \tau_i}. \tag{5.7}$$

Maximum expansion rate τ_i. Some materials such as memory foam expand slowly. Deformation in such materials cannot be determined quasi-statically. But, for situations relevant to rigid-objects simulations, slow expansion is usually not relevant because most objects are capable of expanding faster than they typically separate.

In PFC I assume that objects expand quickly, and that the PFC contact surface can be determined quasi-statically. But, for the purposes of calculating a damping constant, I assume that all compliant materials have a hypothetical maximum expansion rate τ_i that has dimensions of 1/time. If all contact forces were suddenly removed, I assume that the strain in a material decreases at a constant rate τ_i.

Characteristic length L_i. As explained in Appendix C.3, I calculate characteristic thickness as a weighted average of the characteristic length L_{tet} of each tet as follows:

$$L_i \equiv \frac{\sum L_{tet_j} w_j}{\sum w_j} \text{ where} \tag{5.8}$$

$$w_j = \int \|\nabla e\| \, dV_{tet_j}. \tag{5.9}$$

5.2.2 Normal dissipation

Here, I develop a model for the normal dissipation pressure p_χ that opposes the relative *normal* velocity between objects A and B (i.e., opposes $\boldsymbol{v}_{B/A} \cdot \hat{\boldsymbol{n}}$). The total pressure p depends on the nominal pressure field value p_0 and this normal

dissipation pressure p_χ as follows:

$$p = \max(p_0 + p_\chi, 0). \tag{5.10}$$

The max function is to prevent suction (adhesion at the surface). To model adhesion, the zero could be replaced with a negative number. I make the normal damping term p_χ proportional to p_0. The resulting expression for p_χ is shown below:

$$p_\chi = -p_0 \, \chi \, (\boldsymbol{v}_{B/A} \cdot \hat{\boldsymbol{n}}), \tag{5.11}$$

where χ is a combined damping constant, described in subsequent paragraphs. This damping constant has dimensions of inverse velocity. Plugging the above expression for p_χ into (5.10) and rearranging gives us the following equation:

$$p = p_0 \max(1 - \chi \, (\boldsymbol{v}_{B/A} \cdot \hat{\boldsymbol{n}}), 0). \tag{5.12}$$

This equation for p is a continuous function of both time and state, even as contacts are made and broken. Contacts begin and end at pressure field boundaries where p_0 is zero.

The combined damping constant χ. For rigid-compliant contacts, χ is the damping constant of the compliant object. For example, if A is compliant and B is rigid, χ is χ_A. Similarly, if A is rigid and B is compliant, χ is χ_B.

For compliant-compliant contacts, I calculate the combined damping constant χ using the average compliance of each object. To do this, I determine c_A and c_B, the average compliances of A and B, using the formulae:

$$c_A = \frac{L_A}{M_{0A}} \text{ and } c_B = \frac{L_B}{M_{0B}}, \tag{5.13}$$

where M_0 is the nominal scaling factor introduced in §5.1, and L is the average length introduced in §5.2.1. As derived in Appendix C.4, the formula for χ is

$$\chi = \frac{\chi_A c_A^2 + \chi_B c_B^2}{(c_A + c_B)^2}, \tag{5.14}$$

where χ_A and χ_B are determined using $v_{\hat{\boldsymbol{n}}_A}^{max}$ and $v_{\hat{\boldsymbol{n}}_B}^{max}$ (which, in turn, are determined using (5.6)).

Algorithm 1 ComputeContactSurfaceAndTractionCache(A, B) calculates the PFC contact surface $\mathcal{S}$ and information to calculate wrenches $\mathcal{C}$ for objects A and B. For each quadrature point on $\mathcal{S}$, $\mathcal{C}$ contains a tuple with: (1) surface normal $\hat{\boldsymbol{n}}$, (2) point vector $\boldsymbol{r}$, (3) pressure p, and (4) effective area dA. Quantities are expressed in the frame of B unless otherwise noted.

function ComputeContactSurfaceAndTractionCache(A, B)
  $\mathcal{C} \leftarrow \emptyset$ ▷ Initialize traction cache
  $\mathcal{S} \leftarrow \emptyset$ ▷ Initialize PFC contact surface
  $\chi \leftarrow$ calculate damping constant with (5.14)
  $T \leftarrow$ pairs of tetrahedra from A and B flagged for possible intersection in a broad phase
  for all $T_A^i, T_B^i \in T$ **do**
    $\boldsymbol{p}_{0_A}^i \leftarrow$ pressure field from T_A^i
    $\boldsymbol{p}_{0_B}^i \leftarrow$ pressure field from T_B^i
    $\boldsymbol{n}, s_{b/a} \leftarrow$ equilibrium pressure plane from (C.6) using $\boldsymbol{p}_{0_A}^i$ and $\boldsymbol{p}_{0_B}^i$.
    if $\boldsymbol{n} \neq 0$ **then** ▷ see §C.2
      $\hat{\boldsymbol{n}} \leftarrow$ normalize $\boldsymbol{n}$ to calculate the plane normal pointing into B
      $P \leftarrow T_A \cap T_B \cap$ plane given by $\hat{\boldsymbol{n}}$ and $s_{b/a}$ ▷ Compute intersecting polygon
      if $P \neq \emptyset$ **then**
        $\mathcal{S} \leftarrow \mathcal{S} \cup P$ ▷ Update PFC contact surface
        create a point $\boldsymbol{x}_c$ at the centroid of P
        $\mathcal{T} \leftarrow$ tessellate P into triangle set $\mathcal{T}$ by connecting all edges of P to $\boldsymbol{x}_c$ ▷ See Fig. 5.1
        for all $\mathcal{T}_j \in \mathcal{T}$ **do**
          for all quadrature point Q_k and weight $\mathcal{W}_k$ **do**
            $\boldsymbol{r} \leftarrow$ calculate Cartesian position of Q_k
            $p_0 \leftarrow$ calculate pressure field value with (C.4)
            $p \leftarrow$ calculate pressure with (5.18)
            $dA \leftarrow$ multiply $\mathcal{W}_k$ by the area of $\mathcal{T}_j$
            $\mathcal{C} \leftarrow \mathcal{C} \cup (\hat{\boldsymbol{n}}, \boldsymbol{r}, p, dA)$ ▷ Update traction cache for Q_k from $\mathcal{T}_j$
          end for
        end for
      end if
    end if
  end for
  return $\mathcal{C}$, $\mathcal{S}$
end function

5.2.3 Friction

Here I consider Coulomb friction and how to implement it with PFC. Recall that $\boldsymbol{v}_{\text{slip}}$ denotes the relative tangential velocity between the objects at contact points. And that p denotes the total pressure at the PFC contact surface, which includes contributions from the pressure field p_0 and normal damping p_χ. Isotropic Coulomb friction describes a relationship between $\boldsymbol{v}_{\text{slip}}$, p, and a Coulomb friction constant μ as follows:

$$\|\boldsymbol{t}_f\| \leq \mu p \qquad \text{if } \boldsymbol{v}_{\text{slip}} = \boldsymbol{0} \quad \text{(sticking)} \qquad \text{or} \tag{5.15a}$$

$$\boldsymbol{t}_f = -\mu p \hat{\boldsymbol{v}}_{\text{slip}} \quad \text{if } \boldsymbol{v}_{\text{slip}} \neq \boldsymbol{0} \quad \text{(sliding).} \tag{5.15b}$$

The above set of equations is known to be problematic *in the context of rigid contact*. For example, in certain rigid contact situations, there may be multiple $\boldsymbol{t}_f$ that satisfy the above equations, while in others there may be none. PFC is a compliant contact model, so one might expect it to be unproblematic to use Coulomb friction with PFC, but this isn't so. As described earlier, PFC is only compliant in the normal direction. It has no tangential compliance, so some issues with rigid-contact and Coulomb friction (e.g., indeterminacy) still apply.

In continuous-time rigid-object simulations, it's common to approximate Coulomb friction by making friction forces a continuous function of velocity. This approach can be applied to PFC. For example, I use a speed-dependent friction law that is linear viscous up to a cutoff speed v_c and speed independent above v_c. Using this approach, the frictional traction $\boldsymbol{t}_f$ is the product of a pressure

independent quantity $\bar{\boldsymbol{t}}_f$ and pressure p as follows:

$$\boldsymbol{t}_f = p \underbrace{\frac{-\mu \boldsymbol{v}_{\text{slip}}}{\max(\|\boldsymbol{v}_{\text{slip}}\|, v_c)}}_{\bar{\boldsymbol{t}}_f}, \text{ where} \tag{5.16}$$

$$\boldsymbol{v}_{\text{slip}} \equiv \boldsymbol{v}_T \text{ (true in the absence of tangential compliance).} \tag{5.17}$$

This model approaches Coulomb friction in the limit as $v_c \to 0$. Alternatives to (5.16) sometimes create smooth relationships between friction and velocity using sigmoid-like functions such as $\boldsymbol{t}_f = p \tanh(\|\boldsymbol{v}_{\text{slip}}\|/v_c)$.

Speed-dependent approximations of rate-independent Coulomb friction eliminate indeterminacies, but can introduce artifacts. For example, they don't eliminate sliding, even for small tangential forces. A block on an inclined ramp will always slide down, even if the ramp's slope is arbitrarily small. In robotics, this unavoidable creeping is a problem when attempting to, e.g., obtain static initial conditions for stacked objects.

To model static situations with finite friction forces, one can use a contact model with both normal and tangential compliance. Or add tangential compliance to a model that already has normal compliance. For these types of contact models, the equations for Coulomb friction in (5.15) are not problematic. For the purposes of using this type of friction law, I describe a tangential compliance model for PFC in Chapter 6.

5.2.4 Traction with damping and friction

For the modeling choices in this chapter, I provide a compact expression for $\boldsymbol{t}$. Substituting p_χ from (5.11) into p from (4.5) allows us to calculate p at Q as

follows:

$$p = p_0 \max(1 + \chi(\boldsymbol{v}_{B/A} \cdot \hat{\boldsymbol{n}}), 0). \tag{5.18}$$

Substituting the expression for $\boldsymbol{t}_{\hat{\boldsymbol{n}}}$ from (4.8) and $\boldsymbol{t}_f$ from (5.16) into $\boldsymbol{t}$ from (4.3) produces the following:

$$\boldsymbol{t} = p(\hat{\boldsymbol{n}} + \bar{\boldsymbol{t}}_f). \tag{5.19}$$

5.2.5 Asymptotic Time Complexity Analysis

The time complexity of this contact model follows from the cost of recursively comparing two bounding volume hierarchies; other operations (e.g., polygon clipping) scale linearly with the number of intersecting tets. Suppose that A has n_A tets and B has n_B tets, of which m intersect. The expected time complexity of collision detection is $O(m \lg(n_A + n_B))$ for a bounding volume hierarchy[3]. In the worst case, m is $O(n_A n_B)$ and corresponds to intersecting combs (i.e., a Chazelle polyhedron) as Fig 1.4 of [38] shows. In practice, the cost of recursive hierarchy traversal is usually small, as most meshes are not degenerate in this way.

5.3 Dynamics demonstrations

This section shows multibody dynamics scenarios simulated with the PFC implementation described here. These demonstrations were simulated in Julia using the RigidBodyDynamics.jl [31] library. They were integrated with the continuous-time implicit integrator Radau described in [23].

[3]The depth of a balanced tree with n_A leaves is $O(\lg n_A)$. Therefore, $O(\lg(n_A + n_B))$ intersection tests need to be performed against each object.

5.3.1 Pencil grip

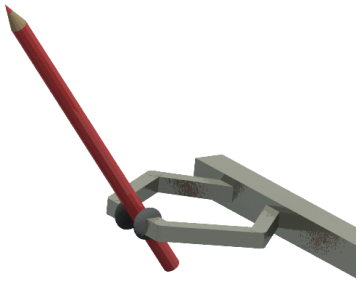

Figure 5.3: **Reorienting a pencil in a pinched grip.** The pencil rotates without falling when the grip loosens during the task in §5.3.1.

I used a gripper to manipulate a pencil to demonstrate a task that relies on area-dependent scrubbing torques. The gripper makes the pencil rotate without falling by loosening the pinch (see Fig. 5.3). Tightening the grip causes rotation to stop. Most point-contact models will fail to model this task correctly because most do not create rotational scrub torques, and those that do usually assume a constant torsional stiffness (e.g., MuJoCo). This demonstration is in the video titled 'pencil.mp4'.

5.3.2 Cam Cleat

I simulate a cam cleat to demonstrate the ability of PFC to simulate mechanisms whose behavior depends on the interaction of distributed frictional contacts over complex non-convex geometry. A cam cleat allows a rope to travel in a single direction, and can be used when raising the sails of boats. Figure 5.4 displays images of the cam cleat simulation. I used a rubber-coated rod instead of a rope because rigid objects don't stretch. The rod moves freely forward, but is prevented from moving backwards by the cam cleat. This demonstration is in the video titled 'cam_cleat.mp4'.

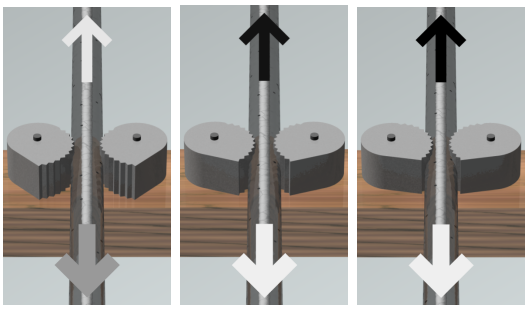

Figure 5.4: **Simulation of a cam-cleat.** Darker arrows denote larger forces. **Left:** a rubber-coated cable initially travels downward due to a small applied force. The rubber slips against the spinal. **Center:** an upward force is applied; the cable moves upward; the spinal moves with the rubber. **Right:** the upward force remains constant; the cable does not move.

5.3.3 Sorting

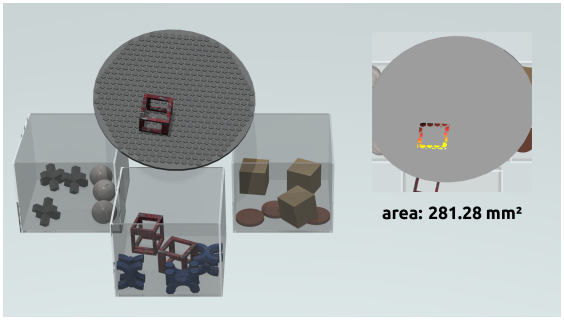

Figure 5.5: **Visualization of the sorting scenario described in §5.3.3.** A sensing and tipping paddle is placed above three bins. The paddle calculates the contact surface area from PFC data as shown on the right. Depending on the size of the sensed contact surface area, the paddle tilts toward one of the three bins (see text).

When designing controllers in simulation, physics-engines are used to simulate both the forward dynamics *and* the sensor measurements used by controllers. Simulated sensor measurements tell the controller what's going on inside the physics-engine. Typical rigid-object physics engines can simulate joint position sensors, joint velocity sensors, and force sensors. But, the typical rigid-object physics engine cannot simulate certain sensors, including tactile sensors. The inability to simulate tactile data limits the utility of the typical rigid-object simulator to de-

sign controllers that use such data.

Virtual tactile data is potentially useful for control, state estimation, and machine learning applications. This scenario shows how PFC modeled contact surface information can be used as virtual tactile data. In this example, objects are sorted into one of three bins based on PFC modeled contact area (see Fig. 5.5). The right panel of this figure shows the PFC contact surface and pressure distribution created by our model. Lighter colors indicate higher pressure. The controller in this task used only the PFC modeled contact surface geometry to calculate area. This demonstration is in the video titled 'sorting.mp4'.

5.3.4 Bolt

I simulate a bolt being aligned with and then screwed into a threaded hole. Figure 5.6 displays images of this simulation. In this simulation, gravity does not act on the bolt. A gravity-like force pushes the bolt to the left. The bolt has 6 kinematic DoF, and is prevented from tipping by the bracket's sidewall. Initially, an external torque turns the bolt counterclockwise. This would typically loosen the bolt, but because the bolt isn't engaged, the threads slip past each other once per turn. Just after the threads jump down one pitch length, the bolt and hole are aligned. The external torque than changes direction to turn the bolt clockwise to tighten. This demonstration starts at the 1:05 mark in the video titled 'pfc_video.mp4'.

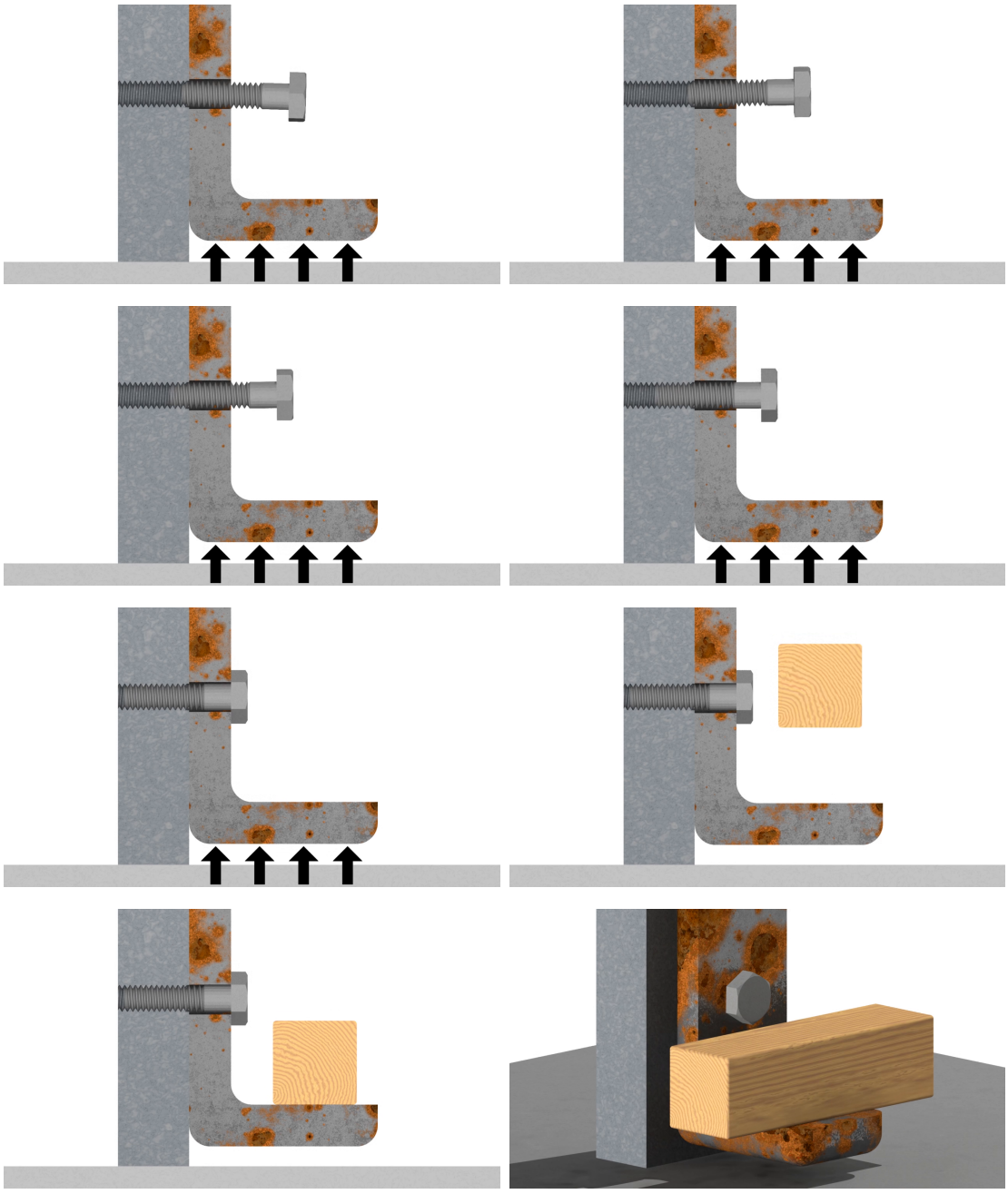

Figure 5.6: **Threading a bolt.** Cut-away view of a bolt being aligned with and threaded into a hole. As explained in 5.3.4, the bolt is turned counterclockwise until the bolt and hole are aligned. Once it's aligned, the bolt is turned clockwise to tighten.

5.3.5 Comments on demonstrations

The pencil grip scenario in §5.3.1 and the cam cleat scenario in §5.3.2 were fast to simulate with this PFC implementation. For these two simulations, it took one second for a computer to simulate each second in simulation. The sorting scenario in §5.3.3 and the bolt scenario in §5.3.4 were slow to simulate. Each of these two simulations took about four hours. The sorting scenario was slow to simulate because the stacked items in the bins slid on top of each other, resulting in stiff dynamics. The bolt scenario was slow to simulate for two reasons. First, the thread contains ≈ 92 triangles per rotation, so the number of tet-tri contacts became large as the bolt was screwed in. Second, the geometry of the bolt and the hole were discrete, so the screwing motion, instead of being a smooth motion, was a sequence of perhaps tens of thousands of discrete collisions. These collisions caused the error controlled integrator to take small time steps.

CHAPTER 6

PFC (TANGENTIAL COMPLIANCE)

In this chapter, I describe a tangential compliance model for PFC. This model is useful at contacts, where modeling tangential compliance is of direct interest. This model is also useful because it allows one to use the strict discontinuous version of Coulomb friction in (5.15). Using a non-regularized formulation of friction allows one to e.g., find static initial conditions for stacked objects. This model is not needed if neither of these two features is desired.

Tangential compliance for a point. In Chapter 6.1, I describe the tangential compliance model for a point contact from [17]. This model has two state variables, one for the tangential deformation in each direction. This chapter begins with a point contact because this model is not widely known.

I then provide an explicit formula for the tangential friction produced by this model for the case of Coulomb friction. The derivation for this formula consists of a single step: find $\hat{\boldsymbol{v}}_{\text{slip}}$. I find $\hat{\boldsymbol{v}}_{\text{slip}}$ in a way that doesn't require knowing $\boldsymbol{v}_{\text{slip}}$. Some readers find this approach unsatisfying, which makes this approach controversial. Finally, I provide examples of how this tangentially compliant point contact with Coulomb friction behaves in 1D and then 2D.

Tangential compliance for a contact surface. For PFC, one could add tangential compliance to a contact surface by using the pointwise model described above at all points on *the boundary* of one of the objects. This approach adds an infinite number of deformation state variables, which is okay in the context of a continuous model. But it's not straightforward to turn it into a discrete model. In a discrete model, one needs to integrate the deformation of a finite number of

variables. Chapter 6.2 describes a practical and strictly physical way to integrate this deformed continuum as a 6-DoF spatial[1] vector.

6.1 Tangential compliance and Coulomb friction for a point contact

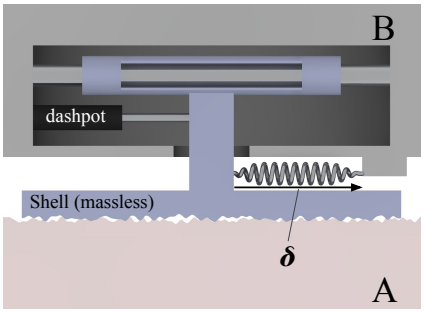

Figure 6.1: **A compliant contact.** Frictional contact between objects A and B is mediated by a "shell" (blue) of negligible mass. The shell is attached to B by a frictionless rail that prevents rotation and normal displacement. The tangential displacement allowed by the rail is resisted by a spring and damper in parallel. The sliding of the shell against A is subject to friction from the pressure p. In short, tangential contact between A and B occurs through a spring and dashpot in parallel (Kelvin-Voigt), in series with friction.

A massless 'shell' is attached to object B by a parallel spring-damper (see Fig. 6.1). The spring constant is k. The damping constant is c. A friction traction $\boldsymbol{t}_f$ (e.g., from Coulomb friction, but not necessarily from Coulomb friction) is applied between the shell and A. The displacement of B relative to the shell is $\boldsymbol{\delta}$ (see Fig. 6.1). Kinematics says that the stretch rate $\dot{\boldsymbol{\delta}}|_k$ is the relative tangential

[1]Rigid-object dynamics contains both rotational and translational aspects. For example, velocity has a rotational component (i.e., angular velocity) and a translational component (e.g., linear velocity). These rotational and translational components are often grouped together into 'spatial' quantities. For example, a spatial velocity vector contains an angular velocity (3 DoF) and a translational velocity (3 DoF). Similarly, a spatial force contains a torque (3 DoF) and a force (3 DoF). A spatial velocity is sometimes called a twist, while a spatial force is sometimes called a wrench.

velocity of A and B, i.e., $\boldsymbol{v}_T$ minus the slip velocity $\boldsymbol{v}_{\text{slip}}$ of the shell of B relative to object A:

$$\dot{\boldsymbol{\delta}}|_k = \boldsymbol{v}_T - \boldsymbol{v}_{\text{slip}}. \tag{6.1}$$

I denote the stretch rate with $\dot{\boldsymbol{\delta}}|_k$ instead of $\dot{\boldsymbol{\delta}}$, as k is non-constant later. In effect, a spring-damper attached to a massless shell mediates tangential contact between A and B.

Tangential tractions applied *to* the shell are Coulomb friction $\boldsymbol{t}_f$ from A, and the spring-damper forces $k\boldsymbol{\delta}$ and $c\dot{\boldsymbol{\delta}}|_k$ from B. The laws of statics apply within this model [17], meaning that $\mathbf{0} = \boldsymbol{t}_f + k\boldsymbol{\delta} + c\dot{\boldsymbol{\delta}}|_k$. It follows that $\boldsymbol{t}_f$ is:

$$\boldsymbol{t}_f = \underbrace{-k\boldsymbol{\delta}}_{\substack{\text{spring traction} \\ \text{applied to } B}} + \underbrace{-c\dot{\boldsymbol{\delta}}|_k}_{\substack{\text{damper traction} \\ \text{applied to } B}}. \tag{6.2}$$

Plugging the expression for $\dot{\boldsymbol{\delta}}|_k$ in (6.1) into (6.2) gives:

$$\boldsymbol{t}_f = \underbrace{\boldsymbol{t}_s}_{-k\boldsymbol{\delta} - c\boldsymbol{v}_T} + c\boldsymbol{v}_{\text{slip}}, \text{ where} \tag{6.3}$$

$\boldsymbol{t}_s$ is the traction when a contact is sticking, i.e., when $\boldsymbol{v}_{\text{slip}}$ is zero. The stretch-rate of this constant stiffness spring $\dot{\boldsymbol{\delta}}|_k$ is

$$\dot{\boldsymbol{\delta}}|_k = -\tau^{-1}\left(k^{-1}\boldsymbol{t}_f + \boldsymbol{\delta}\right), \tag{6.4}$$

where τ is $-k/c$. Equations (6.3) and (6.4) apply irrespective of the shell sliding against A (i.e., whether $\|\boldsymbol{v}_{\text{slip}}\| < 0$ or $\|\boldsymbol{v}_{\text{slip}}\| = 0$). They are also independent of the friction model.

6.1.1 Closed-form solution for Coulomb friction

In this paragraph, I give the closed-form equation for $\boldsymbol{t}_f$ for the Coulomb Friction model, and explain how to derive it. For Coulomb friction, the closed-form equation

for $\boldsymbol{t}_f$ is

$$\boldsymbol{t}_f = \left\{ \begin{array}{ll} \boldsymbol{t}_s & \text{if } \|\boldsymbol{t}_s\| \leq \mu p \\ \hat{\boldsymbol{t}}_s \mu p & \text{if } \|\boldsymbol{t}_s\| > \mu p \end{array} \right\}, \begin{array}{l} \text{(sticking)} \\ \text{(sliding)} \end{array} \tag{6.5}$$

$$\text{where} \quad \boldsymbol{t}_s \equiv -k\boldsymbol{\delta} - c\boldsymbol{v}_T. \tag{6.6}$$

These equations for $\boldsymbol{t}_f$ make $\boldsymbol{t}_f$ as close to the sticking traction $\boldsymbol{t}_s$ as Coulomb friction allows. Conceptually, (6.5) minimizes slip velocity within the limits of Coulomb friction. To arrive at this solution, recall the definition of Coulomb friction from (5.15) copied below:

$$\begin{array}{rlll} \|\boldsymbol{t}_f\| \leq \mu p & \text{if } \boldsymbol{v}_{\text{slip}} = \boldsymbol{0} & \text{(sticking)} & \text{or} \\ \boldsymbol{t}_f = -\mu p \hat{\boldsymbol{v}}_{\text{slip}} & \text{if } \boldsymbol{v}_{\text{slip}} \neq \boldsymbol{0} & \text{(sliding)} & \end{array}$$

The sticking case of (6.5) is determined from (6.3). If the slip velocity is zero, then $\boldsymbol{t}_f = \boldsymbol{t}_s$. For the sliding case of (6.5), I need to know the slip direction. To determine the slip direction $\hat{\boldsymbol{v}}_{\text{slip}}$, I plug the expression for $\boldsymbol{t}_f$ in (6.3) into the sliding case of (5.15b) copied above and rearrange to get:

$$\boldsymbol{t}_s = -\mu p \hat{\boldsymbol{v}}_{\text{slip}} - c\boldsymbol{v}_{\text{slip}}. \tag{6.8}$$

From here I can deduce that $\hat{\boldsymbol{v}}_{\text{slip}} = -\hat{\boldsymbol{t}}_s$. The sliding case from (6.5) follows. To familiarize readers with this model, I show how it behaves in 1D and 2D in §6.1.2 and §6.1.3 respectively.

6.1.2 Example: Dragging, 1D slip

The friction force history for the spring-damper mediated model in Fig. 6.1 for a simple relative-motion history is shown in Fig. 6.2. Object B is stationary before

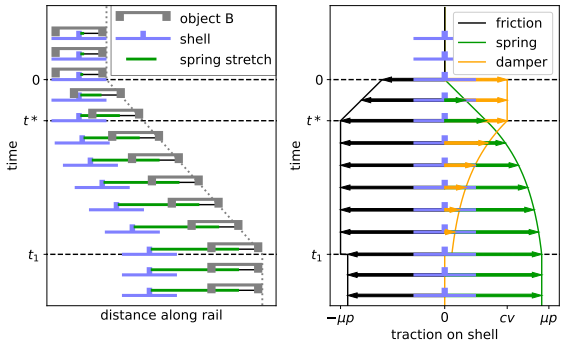

Figure 6.2: **1D illustration of the sliding history effects introduced by the spring-damper mediated friction.** Object B is initially stationary and there is no spring stretch. Then, starting at $t = 0$, B moves to the right at constant speed. B stops at t_1. Time increases from top to bottom. **Left:** Cartoon of object B, the shell, and the spring stretch at various times. **Right:** Spring, damping, and friction tractions on the *shell* at these same times. Note that the sum of the spring, damper, and friction tractions is zero. The damper and friction forces both jump when motion starts at $t = 0$. The velocity in this example is not enough to induce immediate slip. The spring force increases linearly with time until the maximum friction magnitude (i.e., the friction strength μp) is reached at t^*. The sum of the spring-damper traction remains at μp as traction is gradually redistributed from the damper to the spring as the stretch increases and the stretch rate decreases (which causes $\boldsymbol{v}_{\text{slip}}$ to increase since $\dot{r}_\perp$ is constant in this example). When the overall motion stops at t_1, the friction force drops by the magnitude of the still remnant force in the damper. As shown, the locked-in friction force at the end of the relative motion of B and A is close to the frictional strength.

$t = 0$, has a constant velocity v from $t = 0$ to t_1, and is stationary again after t_1. When B moves, the shell initially sticks, then begins to slide at time t^* when the traction reaches the friction strength μp. It then sticks again at t_1 with a residual friction traction. The figure shows the change in the friction, spring, and damper forces in this motion.

6.1.3 Example: Dragging, 2D slip

The spring-damper model in §6.1.1 introduces history effects illustrated here with a simple transient motion. Assume a constant normal pressure p between objects

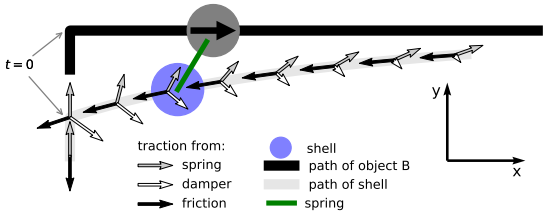

Figure 6.3: **The spring-damper model introduces a history dependence of the friction force.** Here, the directional aspects are illustrated, by considering a special relative motion of objects B and A (6.1.3). A is stationary; only the coordinate system of A is shown here. B has been moving steadily in the positive y direction at constant speed so that the damper applies no traction. B follows the dark line at a constant speed. The shell's motion follows the light line. The evolution of the spring, damper and friction forces are shown with the arrows. These are drawn at points along the trajectory of the shell, which is depicted in light gray. At the discontinuity at $t = 0$, the damper and friction forces suddenly change but the spring stretch, and by extension, the spring force, do not. As time progresses, the damping force relaxes towards zero, and the friction force is asymptotically carried by the spring alone.

A and B. Object B travels in the y direction for a long time and then has a sudden transition at $t = 0$ in the x direction, for $0 < t$, maintaining a constant speed at all times. Figure 6.3 shows the paths of object B and the shell. Tractions applied to the *shell* by the spring, viscous sliding, and Coulomb friction are plotted at various points during this motion.

The displacement rate is zero when object B is traveling in the y direction, which corresponds to a steady-state scenario. I can see this in Fig. 6.3 because the damping traction arrow has zero length before $t = 0$. This steady-state behavior ends at $t = 0$; the positions of object B and the shell at this time are marked in the figure. For $t > 0$, object B moves in the x direction which causes the shell to slowly travel in the y direction relative to object B. The traction arrows demonstrate the following features of this model: (1) the frictional traction opposes the velocity of the shell (friction arrows pointing in the opposite direction of the path), (2) tractions applied to the shell from the spring, viscous sliding, and Coulomb friction

sum to zero at all times, (3) the spring traction varies continuously at all times because the spring displacement is a continuous function of state, and (4) the spring traction always acts in the direction of the spring, while the damping traction *does not*.

6.2 Tangential compliance and friction at a distributed contact

As described earlier, one way to model the tangential compliance of a continuous contact is to use a pointwise tangential compliance model at all points on the boundary of one of the objects. The problem with numerically implementing this model is that it requires integrating an infinite number of deformation state variables. In this section, I develop a spatial[2] approximation of this continuous deformation.

There are two main observations used to arrive at the spatial discretization of tangential deformation. The first observation is that on average, the deformation of points not actively engaged in contact tends to be small. In the absence of contact, the tangential deformation decays with the characteristic time of the spring-damper system τ. This characteristic time is small compared to the average time since last contact. For this reason, I assume that the deformation at these points is zero.

[2]As noted in §6.1, rigid-object dynamics contains both rotational and translational aspects. For example, velocity has a rotational component (i.e., angular velocity) and a translational component (e.g., linear velocity). These rotational and translational components are often grouped together into 'spatial' quantities. For example, a spatial velocity vector contains an angular velocity (3 DoF) and a translational velocity (3 DoF). Similarly, a spatial force contains a torque (3 DoF) and a force (3 DoF). A spatial velocity is sometimes called a twist, while a spatial force is sometimes called a wrench.

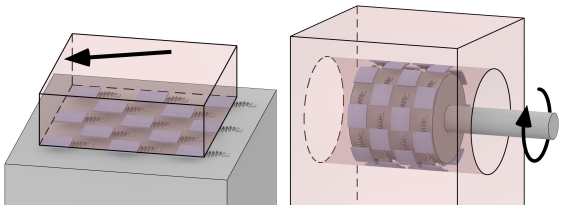

Figure 6.4: **Constant magnitude spring displacement caused by rigid-object motion.** Shells are either all sliding or all sticking. **Left:** Object A are B are blocks. A translates relative to B. **Right:** Object A is a block with a cylinder bored into it. Object B is a piston comprising a cylinder (gray) attached to a pole (light gray). B rotates inside A; gravity is negligible.

The second observation is that the deformation of points actively involved in contact is largely the result of relative spatial motion. During sticking, relative object motion is responsible for all relative motion at the PFC contact surface. This is also true during sliding for certain contacts, including those in Fig. 6.4. For these reasons, I model the deformation of all points on the contact surface with a single spatial deformation.

Before I describe this spatial approach in detail, it's important to clarify one thing. The discrete model is only concerned with how spring displacements are propagated. It does not affect how tractions/wrenches are calculated at each instant.

6.2.1 Spatial deformation

To capture displacement caused by rigid motion, I parameterize $\boldsymbol{\delta}$ as the sum of (1) a translation $\boldsymbol{\Delta}_{\boldsymbol{r}}$, and (2) a rotation $\boldsymbol{\Delta}_{\theta}$ about the pressure-weighted PFC contact surface center $\boldsymbol{c}$

$$\boldsymbol{c} = \frac{\int \boldsymbol{r} p \, \mathcal{S}}{\int p \, \mathcal{S}}. \tag{6.9}$$

The vector $\boldsymbol{r}$ gives the position of point Q on the PFC contact surface relative to $\boldsymbol{c}$. The displacement at Q is

$$\boldsymbol{\delta} = (\boldsymbol{\Delta}_\theta \times \boldsymbol{r} + \boldsymbol{\Delta}_{\boldsymbol{r}}) \perp \hat{\boldsymbol{n}}. \tag{6.10}$$

To propagate the reduced dimension state $\boldsymbol{\Delta}$, I first note that the friction wrench has a spring and a damper contribution. The spring contribution $\boldsymbol{f}_{\boldsymbol{\Delta}}$ is a function of a stiffness matrix $\boldsymbol{K}$ as follows:

$$\boldsymbol{f}_{\boldsymbol{\Delta}} = -\boldsymbol{K}\boldsymbol{\Delta}, \tag{6.11}$$

$$\text{where } \boldsymbol{\Delta} \equiv \begin{bmatrix} \boldsymbol{\Delta}_\theta \\ \boldsymbol{\Delta}_{\boldsymbol{r}} \end{bmatrix}; \tag{6.12}$$

a result derived in Appendix D.1. The displacement rate of this constant stiffness model is $\dot{\boldsymbol{\Delta}}|_{\boldsymbol{K}}$. I denote the displacement rate with $\dot{\boldsymbol{\Delta}}|_{\boldsymbol{K}}$, instead of $\dot{\boldsymbol{\Delta}}$, as $\boldsymbol{K}$ is non-constant later. The expression for $\dot{\boldsymbol{\Delta}}|_{\boldsymbol{K}}$ is simplest in a fixed frame C that is instantaneously coincident with $\boldsymbol{c}$ and aligned with the frame of B. I use the superscript ^{C} to denote quantities expressed in frame C. As derived in Appendix D.2, $\dot{\boldsymbol{\Delta}}|_{\boldsymbol{K}}$ is

$$\dot{\boldsymbol{\Delta}}|^C_{\boldsymbol{K}} = -\tau^{-1} \left(\boldsymbol{K}^C\right)^{-1} \left(\boldsymbol{f}^C_{\boldsymbol{\Delta}} - \boldsymbol{f}^C_c\right). \tag{6.13}$$

Equation (6.13) allows us to propagate spring displacements for a constant stiffness surface using $\boldsymbol{f}_c$ as shown:

$$\boldsymbol{f}_c = \int \begin{bmatrix} \boldsymbol{r} \times \boldsymbol{t}_f \\ \boldsymbol{t}_f \end{bmatrix} d\mathcal{S}. \tag{6.14}$$

The friction traction $\boldsymbol{t}_f$ is calculated with a user-chosen pointwise friction model such as (6.5) or (5.16). During sliding, this model integrates PFC contact surface displacements consisting of rigid motion (see Fig. 6.4) exactly, and behaves sensibly when they don't (see Appendix 6.2.2).

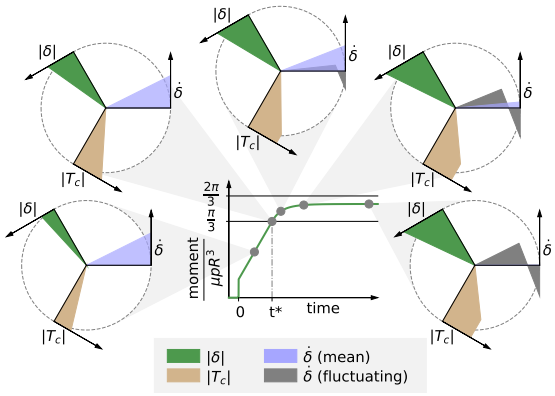

Figure 6.5: **Slipping in the distributed brush model. Middle:** At $t = 0$, the moment increases from zero to oppose the cylinder's rotation. This moment increases linearly after $t = 0$ until t^*, at which point slip begins. **Outer plots:** The dashed circle represents the PFC contact surface. Each plot shows the radial distributions of $\boldsymbol{\delta}$, $\boldsymbol{t}_f$, and $\dot{\boldsymbol{\delta}}$ at the marked time. **Surface sticks (1+2):** The reduced model captures the displacement field exactly. The mean part of $\dot{\boldsymbol{\delta}}$ increases linearly with time; the fluctuating part of $\dot{\boldsymbol{\delta}}$ is zero. **Surface slides (3+4):** $\boldsymbol{t}_f$ saturates starting at the outer edge. The reduced model fails to capture the $\dot{\boldsymbol{\delta}}$ field exactly, and therefore the fluctuating part of $\dot{\boldsymbol{\delta}}$ is non-zero. **Steady-state (5):** The mean part of $\dot{\boldsymbol{\delta}}$ is zero and the fluctuating part of $\dot{\boldsymbol{\delta}}$ is constant.

6.2.2 Example: Cylinder spinning around longitudinal axis

The continuous model and the spatial approximation behave differently in this scenario. Object A is a rigid horizontal half-plane and object B is a compliant cylinder lying on end. I assume that the overlap is small, and therefore the PFC contact surface is a uniform pressure circle. The coefficient of friction between A and B is μ. The cylinder has a radius R. The cylinder is stationary before $t = 0$, and begins rotating at a constant angular velocity when $t = 0$. When B moves, the shell initially sticks, then begins to slide at time t^* when the moment reaches a critical value. The torque associated with a given motion is simply the integral of contributions from traction across the equal pressure contact surface [22].

In the full model, each spring displaces and eventually reaches the steady-state stretch $\mu/\bar{k}$. Springs nearer the edge reach this terminal stretch more quickly. The

steady-state moment from the PFC contact surface is:

$$M_{cont} = \frac{2\pi\mu p R^3}{3}, \tag{6.15}$$

which is also the largest moment that can be applied to the cylinder without it rotating.

In the reduced model, the PFC contact surface sticks until $\|\boldsymbol{t}_f\| = \mu p$ on the outer edge. When sliding begins, the moment is

$$M_{reduced} = \frac{\pi\mu p R^3}{2}. \tag{6.16}$$

Figure 6.5 shows the details of transient behavior. Unlike the full model, the steady-state moment depends on the rotation rate and approaches M_{cont} as the rotation rate approaches ∞. This reduced dimension model is more accurate for PFC contact surfaces with higher pressure on the outside. Table 6.1 shows the rotation resistance accuracy for simple rotationally-symmetric shapes for a linear pressure field.

Contact surface	$M_{reduced}/M_{cont}$	
	decimal	frac.
Thin Ring	1.000	1
Circle	0.750	3/4
Half-sphere	0.679	$32/(15\pi)$
Cone	0.600	3/5

Table 6.1: **Proportion of rotation resistance for simple rotationally-symmetric shapes.**

6.2.3 Adaptive stiffness

The friction model in §6.1 uses a constant stiffness k at each point. If k is small, displacement will be excessive; if k is large, numerical integration will be slow. For

point-contact, one can determine k using the expected normal force. For PFC, this isn't possible, as pressure on the PFC contact surface varies. To ensure consistent behavior for surfaces, I make the stiffness the product of a constant $\bar{k}$ and p as follows:

$$k = \bar{k}p. \tag{6.17}$$

Similarly, I make the damping the product of a constant $\bar{c}$ and p as follows:

$$c = \bar{c}p. \tag{6.18}$$

When k and c are determined with (6.17) and (6.18) respectively, the deformation before sliding occurs depends on μ, and not p.

This approach to adaptive stiffness has an additional advantage when used with the spatial dimension reduction approach described earlier. It promotes agreement with the continuous model. By making sliding not depend on p, it makes a contact surface more likely to be either all sticking or all sliding. For example, the contact surfaces in Fig. 6.4 would still be either all sliding or all sticking, even if the contact surfaces were variable-pressure.

6.2.4 Integration

For use in general simulations, I need to propagate $\boldsymbol{\Delta}$ for variable stiffness PFC contact surfaces. To be physically sensible, this needs to occur in a way that: (1) ensures spring energy changes are caused by work only and (2) is invariant to rotation, translation, and choice of units. The stiffness matrix $\boldsymbol{K}$ changes as the PFC contact surface changes. The spring energy $U_{\boldsymbol{K}}$ depends on $\boldsymbol{K}$ and $\boldsymbol{\Delta}$

according to

$$U_{\boldsymbol{K}} = \frac{1}{2}\boldsymbol{\Delta}^T \boldsymbol{K} \boldsymbol{\Delta}. \tag{6.19}$$

To create a unit invariant state variable, I write $\boldsymbol{K}$ as the product of a diagonal matrix $\boldsymbol{S}$ and a unitless matrix $\bar{\boldsymbol{K}}$.

$$\boldsymbol{K} = \boldsymbol{S}\bar{\boldsymbol{K}}\boldsymbol{S} \tag{6.20}$$

$\boldsymbol{S}$ makes the traces of the top-left (rotational) and bottom-right (translational) blocks of $\bar{\boldsymbol{K}}$ equal. Our state variable $\boldsymbol{s}$ is defined to be

$$\boldsymbol{s} \equiv \bar{\boldsymbol{K}}^{1/2}\boldsymbol{S}\boldsymbol{\Delta}, \tag{6.21}$$

which makes $U_{\boldsymbol{K}} = \frac{1}{2}\boldsymbol{s}^T\boldsymbol{s}$. The power delivered to the spring $W_{\boldsymbol{K}}$ (6.22) is equal to the change in spring energy $\dot{U}_{\boldsymbol{K}}$ (6.23).

$$W_K = \dot{\boldsymbol{\Delta}}|_{\boldsymbol{K}}^T \boldsymbol{K}\boldsymbol{\Delta} \tag{6.22}$$

$$\dot{U}_K = \dot{\boldsymbol{s}}^T\boldsymbol{s} \tag{6.23}$$

Using (6.22) and (6.23), the derivative of $\boldsymbol{s}$ is

$$\dot{\boldsymbol{s}} = \bar{\boldsymbol{K}}^{1/2}\boldsymbol{S}\dot{\boldsymbol{\Delta}}|_{\boldsymbol{K}}. \tag{6.24}$$

Propagating $\boldsymbol{\Delta}$ by integrating $\boldsymbol{s}$ is numerically necessary because $\boldsymbol{\Delta}$ is not bounded as $\boldsymbol{K} \to \boldsymbol{0}$. To simplify $\dot{\boldsymbol{s}}$, I substitute (6.13) into (6.24) to obtain

$$\dot{\boldsymbol{s}} = -\tau^{-1}\bar{\boldsymbol{K}}^{-1/2}\boldsymbol{S}^{-1}\left(\boldsymbol{f}_c - \boldsymbol{f}_{\boldsymbol{\Delta}}\right). \tag{6.25}$$

As contact is lost, $\boldsymbol{K} \to \boldsymbol{0}$, and therefore for energetic consistency, $\|\boldsymbol{\Delta}\| \to \infty$. When $\|\boldsymbol{\Delta}\| \to \infty$, $\|\boldsymbol{f}_c\| \ll \|\boldsymbol{f}_{\boldsymbol{\Delta}}\|$, and when this occurs, the contribution from $\boldsymbol{f}_c$ is $\boldsymbol{0}$ because the contribution from $\boldsymbol{f}_{\boldsymbol{\Delta}}$ is $-\tau^{-1}\boldsymbol{s}$, which is bounded. A well-behaved expression for $\dot{\boldsymbol{s}}$ follows

$$\dot{\boldsymbol{s}} = \left\{ \begin{array}{ll} -\tau^{-1}\left(\bar{\boldsymbol{K}}^{-1/2}\boldsymbol{S}^{-1}\boldsymbol{f}_c + \boldsymbol{s}\right) & \text{if } \boldsymbol{K} \neq \boldsymbol{0} \\ -\tau^{-1}\boldsymbol{s} & \text{otherwise} \end{array} \right\}. \tag{6.26}$$

CHAPTER 7

LIMITATIONS OF PFC

7.1 PFC needs a velocity-stepping implementation

There are two main approaches to integrating rigid-object motion: continuous time and velocity-stepping. In the continuous-time approach, the system's state vector is differentiable. This approach uses a continuous time integrator (e.g., Runge Kutta, implicit Euler, etc.) to propagate the state of the system using derivative information. When the system's motion is smooth and polynomial-like, large time steps (e.g., $0.01s$ for a typical robotics task) are often possible. However, small time steps (e.g., $10^{-8}s$ for a typical robotics task) are typically required when the dynamics contain sudden changes including collisions or contacts that begin to slide.

The need to take small time steps because of sudden changes in dynamics or "events" is a problem for two reasons. First, if the number of events is large, the simulation may take a long time. For example, the threading of a bolt into a hole demonstration described in §5.3.4 took four hours to simulate. The seemingly smooth threading motion contained tens of thousands of collisions due to the use of discrete geometry. Second, needing to take small time steps, even for short periods of time, is unsuitable for real-time applications (e.g., video games, model predictive control). These applications need a way to use a fixed time step, even if multiple events happen during this time.

The velocity-stepping approach is designed to integrate dynamics that is stiff and/or non-smooth. In the absence of contact, velocity-stepping implementations

are often no different than semi-implicit Euler. At each time step, semi-implicit Euler first calculates velocities with explicit Euler, and *then* uses these newly calculated velocities to update the positions. In the presence of contact forces, the velocity-stepping approach changes how it calculates velocities.

The velocity at the end of the time step depends on the net contact impulse (i.e., the integral of the contact forces over the time step). But, when a time step contains events at unknown times, there is no fast sampling-based way to estimate this net impulse (e.g., with quadrature). For this reason, the velocity-impulse approach determines the net impulse in an indirect way by solving an optimization problem at each time step. The problem's unknowns are the net contact impulses and final velocities. Constraints enforce complementary and approximately enforce Coulomb friction. When solved, the internally-consistent answer contains what the dynamics must be.

7.1.1 Creating a velocity-stepping implementation of PFC

The PFC implementation described in Chapters 5 and 6 can be utilized in a continuous time simulation. Additional work is required to port this over to a velocity-stepping simulation.

Existing velocity-stepping simulators treat contact as a pointwise phenomenon. For contacts between two spheres, only one contact point is needed. For a contact between a box and a half-space, four contact points might be used. This approach is efficient when the number of contact points is small because each contact point adds additional constraints.

The naive way to incorporate PFC into a velocity-stepping simulation is to

treat each contact surface quadrature point as a contact point, and then use the approach described above. Although conceptually simple, this approach will be slow in practice. A complex contact surface can have hundreds or thousands of quadrature points.

I think it's possible to create a fast velocity-stepping implementation of PFC by treating the entire contact surface as a single spatial[1] contact. In contrast to point contacts, this distributed contact applies a full 6-DoF wrench. The equations and reasoning that describe how to create the appropriate spatial constraints are missing. Although not directly relevant to velocity-stepping, I believe the reasoning and equations that led to the spatial stiffness matrix in §6 might provide useful inspiration to get one started on this task.

7.2 Credit card problem

PFC determines the contact surface using the object's undeformed geometry. As a consequence, this method predicts extraneous contact surfaces when the penetration depth exceeds object thickness (one object superficially passes through the other). For example, consider the following scenario: a pair of compliant fingers use a pinched grip to pick up a rigid credit card. For a sufficiently tight grip, the undeformed finger geometry passes through the credit card and forms a contact surface on both sides of the card, and with the other finger. One should be aware

[1] As noted in §6.1, rigid-object dynamics contains both rotational and translational aspects. For example, velocity has a rotational component (i.e., angular velocity) and a translational component (e.g., linear velocity). These rotational and translational components are often grouped together into 'spatial' quantities. For example, a spatial velocity vector contains an angular velocity (3 DoF) and a translational velocity (3 DoF). Similarly, a spatial force contains a torque (3 DoF) and a force (3 DoF). A spatial velocity is sometimes called a twist, while a spatial force is sometimes called a wrench.

of this potential issue when using thin geometry.

APPENDIX A

PFC MODEL

A.1 The potential energy of a pressure field contact

Theorem 1. *The formula:*

$$U^{p_0} = \int_{V^\cap} p_0(x, y, z)\ dx\ dy\ dz, \tag{A.1}$$

(a.k.a. Eqn. (4.4)) gives the deformation energy associated with displacing a pressure field.

Proof. Consider the integral of the pressure field p_0 over the displaced volume $V^\cap(t)$. Reynolds transport theorem says that the time derivative of this integral is the sum of the flux of p_0 through the volume's boundary $A^\cap(t)$ and the volume integral of $\dot{p}_0$.

$$\frac{d}{dt}\int p_0\ dV^\cap = \underbrace{\int \dot{p}_0\ dV^\cap}_{\text{change due to } \dot{p}_0} + \underbrace{\int (\boldsymbol{v}\cdot\hat{\boldsymbol{n}}) p_0\ dA^\cap}_{\text{change due to boundary}} \tag{A.2}$$

Without loss of generality, I perform calculations in an object-fixed frame to neglect the "change due to $\dot{p}_0$" term. In this frame, the term associated with the "change due to boundary" is the power delivered by elastic forces.

$$\frac{d}{dt}\int p_0\ dV^\cap = \underbrace{\int \boldsymbol{v}\cdot p_0\hat{\boldsymbol{n}}\ dA^\cap}_{\text{power delivered by elastic forces}} \tag{A.3}$$

In the absence of dissipation, if I take the time integral of (A.3), starting when the intersection volume is zero (i.e., $V^\cap = \emptyset$), I get (4.4). So, the potential energy is the work needed to displace a pressure field:

$$U^{p_0} = \int_{V^\cap} p_0(x, y, z)\ dx\ dy\ dz. \tag{A.4}$$

Note, the integral on the right hand side of (A.4) is a volume integral, and not a path integral. □

A.2 Guarantees of PFC

If pressure fields are continuous, non-negative, and zero on nominal object boundaries, then PFC provides various guarantees.

Guarantees that always apply:

- *Strain energy is a continuous function of state:* See Appendix A.1.
- *Damping doesn't add energy.* In Appendix C.4, I developed an expression for $\dot{u}_d$, the rate at which damping creates energy. This expression contains a negative sign and is quadratic in the normal velocity magnitude. In other words: $\dot{u}_d \propto -(\boldsymbol{v}_{B/A} \cdot \hat{\boldsymbol{n}})^2$.
- *Friction doesn't add energy.* I use $\dot{u}_f$ to denote the rate at which friction creates energy, as follows:

$$\dot{u}_f \equiv \boldsymbol{t}_f \cdot \boldsymbol{v}_{B/A} = \boldsymbol{t}_f \cdot \boldsymbol{v}_T. \tag{A.5}$$

  Substituting the expression for $\boldsymbol{t}_f$ from (5.16) and rearranging shows that $\dot{u}_f$ contains a negative sign and is quadratic in the tangential velocity magnitude. In other words: $\dot{u}_f \propto -\boldsymbol{v}_T \cdot \boldsymbol{v}_T$.

Additional guarantees that apply when $\nabla p_{0A} \neq \nabla p_{0B}$ at each point on $\mathcal{S}$:

- *PFC finds the complete deformed configuration for both objects.* As long as the condition $\nabla p_{0_A} \neq \nabla p_{0_B}$ is met, PFC predicts that every point in the

overlapping region is occupied by the object with the larger pressure field value at that point. Appendix C.2 discusses the probability zero event when $\nabla p_{0A} = \nabla p_{0B}$.

- *PFC finds the complete contact surface.* The PFC contact surface separates deformed objects.
- *A well-defined surface normal exists almost everywhere.*[1]

Additional guarantees that apply when $\nabla p_{0A} \cdot \nabla p_{0B} < 0$ at each point on $\mathcal{S}$:

- *Object deformation and contact wrenches are continuous functions of state:* See Appendix C.5.

[1] In the PFC implementation, the PFC contact surface is polyhedral. The PFC contact surface is ill-defined at the edges, a set of measure zero.

APPENDIX B

PRESSURE FIELDS FOR HALFSPACES

I previously discussed creating pressure fields using the uniaxial compression of an elastic layer as the motivating theory. Approximations based on this compressive assumption tend to be accurate when the following criteria are satisfied: (1) the compressible material is actually layer-like, (2) the contact surface has gently slopes sides, (3) deformable materials have a Poisson's ratio that is not near 0.5, and (4) the characteristic contact radius is much larger than the layer thickness. Pressure fields based on layers may be inaccurate when the layer thickness is large compared to the characteristic contact radius.

When the layer thickness is large compared to the characteristic contact radius, it's possible to create nonlinear pressure fields that give accurate net forces by assuming that the layer is a halfspace. Unlike pressure fields created based on the uniaxial compression of an elastic layer, pressure fields created assuming that the layer is an elastic halfspace depend on the geometry of the penetrating object. To create these geometry-dependent pressure fields, one needs an analytic expression for the contact force between the object and an elastic halfspace.

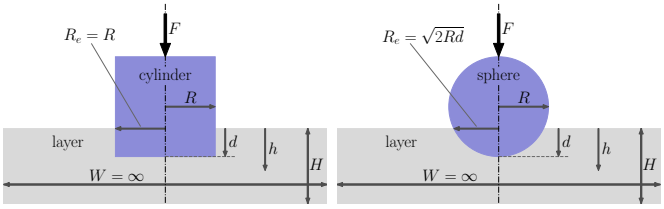

Figure B.1: **Left:** A force F presses rigid objects a distance d into an elastic layer with thickness H. The quantity h denotes distance below the surface. R_e denotes the radius of the PFC contact surface. The vertical dot-dash line in the middle of each schematic denotes rotational symmetry. **Right:** The equation for R_e assumes that penetration is small (i.e., $d \ll R$).

I show how to create pressure fields that agree with elasticity for thick layers for the sphere and the cylinder in Fig. B.1. In these scenarios, a force F presses a rigid object a distance d into a compliant layer (see Fig. B.1). The radius of the PFC contact surface is R_e. The layer has thickness H, elastic modulus E, and Poisson's ratio ν. I assume the contact surface is approximately flat and that the layer thickness is much larger than the contact radius (i.e., $d \ll R_e \ll H$). For the elastic halfspace assumption, the relevant modulus is $E^* = \frac{E}{1-\nu^2}$. The assumptions behind the elastic halfspace approximation are explained in detail in [29].

B.1 Cylinder

As described in [33], the force for the cylinder against a halfspace is:

$$F(halfspace, cylinder) = 2E^* dR. \tag{B.1}$$

The pressure field $p_0(halfspace, cylinder)$ that gives the correct net contact force in (B.1) is:

$$p_0(halfspace, cylinder) = \frac{2E^* h}{\pi R}. \tag{B.2}$$

To verify that the pressure field in (B.1) is correct, I multiply (B.2) by πR^2 (the area of the cylinder's end) which gives $2E^* Rh$, and note that $h = d$ everywhere on the PFC contact surface.

B.2 Sphere

Hertz contact says that the force for the sphere is:

$$F(halfspace, sphere) = \frac{4}{3} E^* R^{1/2} d^{3/2}. \tag{B.3}$$

The pressure field $p_0(halfspace, sphere)$ that gives the correct net contact force in (B.3) is:

$$p_0(halfspace, sphere) = \frac{E^*}{\pi}\sqrt{\frac{h}{R}}. \tag{B.4}$$

To verify that the pressure field in (B.4) is correct, I note the following:

1. The contact scenario is radially symmetric.
2. As in Hertz contact, I approximate the bottom of the sphere with a paraboloid. It follows that for a maximum penetration d, the height at any point on the penetrated sphere is the following function of the radial distance r:

$$h = d - \frac{r^2}{2R}. \tag{B.5}$$

3. The maximum radial distance of points on the contact surface occurs when $h = 0$ and $r = \sqrt{2dR}$.
4. Because the paraboloid approximates only the bottom of a sphere, we can assume that all pressures point upward. This is the assumption made in Hertz contact.
5. Therefore, we want to find a pressure field $p_0(halfspace, sphere)$ that when integrated over the paraboloid gives the total force $F(halfspace, sphere)$ from (B.3) as shown:

$$F(halfspace, sphere) = \int_0^{2\pi}\int_0^{\sqrt{2dR}} r\, p_0(h(r))\, dr\, d\theta \tag{B.6}$$

6. The $p_0(halfspace, sphere)$ that satisfies the equation above is (B.4).

B.3 Force-contact radius relationship for halfspaces

Analytic solutions for forces for halfspaces have a common form. In particular, they take the form of a constant coefficient, times the contact radius R_e, times the maximum penetration d. I show how the force expressions for a cylinder and sphere both follow this pattern.

$$F(halfspace, cylinder) = \underbrace{2E^*}_{\text{coefficient}} \underbrace{R}_{R_e} \underbrace{d}_{\text{penetration}} . \tag{B.7}$$

Similarly, recalling from Fig. B.1 that $R_e = \sqrt{2dR}$, the force expression for a sphere can be written as:

$$F(halfspace, sphere) = \underbrace{\frac{1}{3}E^*}_{\text{coefficient}} \underbrace{\sqrt{2dR}}_{R_e} \underbrace{d}_{\text{penetration}} . \tag{B.8}$$

APPENDIX C

PFC IMPLEMENTATION

C.1 Solving for the PFC contact surface over linear tets

Pressure field for a tet. Within the context of linear interpolation over a tet, scalars and vectors can be written as a linear combination of the values at the four vertices. These linear scaling factors (barycentric coordinates), $\boldsymbol{\zeta} \in \mathbb{R}^4$, are non-negative and sum to one. In order to interpolate pressure field values, we need these weights.

To get these weights, I apply linear interpolation over a tet to the position vector. Let $\boldsymbol{\nu}_{b_1}, \ldots, \boldsymbol{\nu}_{b_4} \in \mathbb{R}^3$ denote the coordinates of the vertices of tet b in object B. A 4×4 matrix, denoted by $\mathbf{X}_b$, transforms the position vector from b in barycentric coordinates to a Cartesian frame, e.g.,

$$\begin{bmatrix} \boldsymbol{r} \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{\nu}_{b_1} & \boldsymbol{\nu}_{b_2} & \boldsymbol{\nu}_{b_3} & \boldsymbol{\nu}_{b_4} \\ 1 & 1 & 1 & 1 \end{bmatrix} \boldsymbol{\zeta}^b = \mathbf{X}_b \boldsymbol{\zeta}^b. \tag{C.1}$$

Inverting the relationship above gives $\boldsymbol{\zeta}^b$ as a function of Cartesian position $\boldsymbol{r}$ as follows:

$$\boldsymbol{\zeta}^b = \mathbf{X}_b^{-1} \begin{bmatrix} \boldsymbol{r} \\ 1 \end{bmatrix}. \tag{C.2}$$

Let $\boldsymbol{p}_{0b}$ denote the pressure field value p_0 at tet b's vertices. The quantity p_0 is a dot product of $\boldsymbol{\zeta}^b$ and the vertex values, i.e., $p_0 = \boldsymbol{p}_{0b} \cdot \boldsymbol{\zeta}^b$. Combining this relationship with (C.2) gives $\tilde{p}_{0b}$, the *interpolated* pressure field value as a function

of position as follows:

$$\tilde{p}_{0b}(\boldsymbol{r}) = \underbrace{\boldsymbol{p}_{0b} \cdot \mathbf{X}_b^{-1}}_{1\times 4 \text{ matrix}} \begin{bmatrix} \boldsymbol{r} \\ 1 \end{bmatrix}. \tag{C.3}$$

Pressure field gradient. The first three terms of the 1×4 matrix in (C.3) are *by definition* the pressure field gradient $\nabla\tilde{p}_{0b}$. The last term is a scalar, denoted by s_b. This term is the extrapolated value of the pressure field at a point that is instantaneously coincident with the origin. Equation (C.3) can be rewritten as follows:

$$\tilde{p}_{0b}(\boldsymbol{r}) = \underbrace{\begin{bmatrix} \nabla\tilde{p}_{0b}^T, \end{bmatrix}}_{1\times 3 \text{ matrix}} \underbrace{\begin{bmatrix} s_b \end{bmatrix}}_{\text{scalar}} \begin{bmatrix} \boldsymbol{r} \\ 1 \end{bmatrix}. \tag{C.4}$$

Contact surface plane. Equation (C.4) also applies to tet a in A:

$$\tilde{p}_{0a}(\boldsymbol{r}) = \begin{bmatrix} \nabla\tilde{p}_{0a}^T, & s_a \end{bmatrix} \begin{bmatrix} \boldsymbol{r} \\ 1 \end{bmatrix}. \tag{C.5}$$

In the PFC model, points on the PFC contact surface have equal pressure field values, so I subtract (C.4) from (C.5) and set $\tilde{p}_{0b} - \tilde{p}_{0a}$ equal to zero to get:

$$0 = \underbrace{\begin{bmatrix} \nabla\tilde{p}_{0b}^T - \nabla\tilde{p}_{0a}^T, \end{bmatrix}}_{\boldsymbol{n}^T} \underbrace{\begin{bmatrix} s_b - s_a \end{bmatrix}}_{s_{b/a}} \begin{bmatrix} \boldsymbol{r} \\ 1 \end{bmatrix}. \tag{C.6}$$

In short, a missing or incomplete PFC contact surface is a probability zero event in practice because of integrator error. For notational brevity, I use $\boldsymbol{n}$ to denote the difference in pressure field gradients and $s_{b/a}$ to denote the difference between s_b and s_a (see (C.6)). The above equation describes a plane; if part of this plane

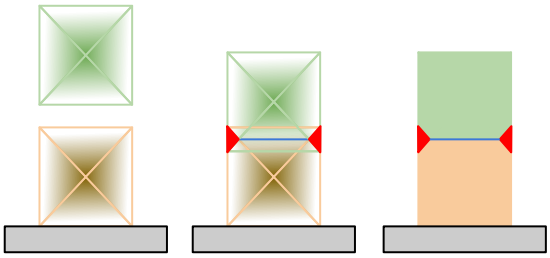

Figure C.1: **An example of the probability zero degenerate situation described in §C.2.** When two identical compliant boxes are dropped directly on top of each other, our implementation omits defining a PFC contact surface through some of the overlapping region. **Left:** The boxes before contact. **Center:** The boxes during contact. The blue line shows the PFC contact surface returned by our implementation. The pressure field values of both objects are exactly identical for all points in the red triangles. Our implementation does not determine the PFC contact surface inside the red regions. **Right:** Same as center figure with grid removed. The deformed configurations for the top box (green) and lower box (orange) calculated by PFC. Our implementation returns no PFC contact surface in the areas demarcated by the red triangles, which lie in the region of overlap.

lies in the tet-tet intersection, this portion makes up part of $\mathcal{S}$. The plane normal, and therefore the PFC contact surface normal $\hat{\boldsymbol{n}}$ is:

$$\hat{\boldsymbol{n}} = \frac{\boldsymbol{n}}{\|\boldsymbol{n}\|}. \tag{C.7}$$

If the gradients of the two tets are identical, $\boldsymbol{n} = 0$, and then $\hat{\boldsymbol{n}}$ is undefined. I discuss this probability zero case in Appx.C.2 (see below).

C.2 Undefined PFC contact surface normal

If the pressure field gradients of two overlapping tets are *exactly* equal, then $\hat{\boldsymbol{n}}$ is undefined. When this occurs, I assume that this tet pair does not contain part of $\mathcal{S}$. Under rare circumstances, this assumption can produce a PFC contact surface that is incomplete or missing altogether. For example, Fig. C.1 shows how dropping two identical compliant boxes exactly on top of each other results in an incomplete PFC

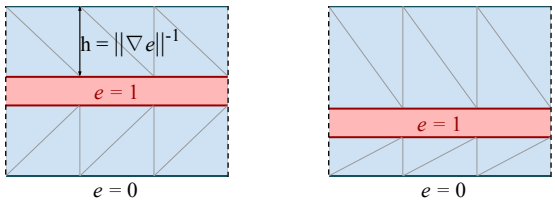

Figure C.2: **Periodic sections of a rigid core (red) embedded in a compliant layer (blue).** I use the way the triangles in the compliant layer change between the left and right panels to create a formula for characteristic distance. **Left:** The thickness of the layer is the same on both sides. **Right:** The rigid rectangle is moved, but the average layer thickness stays the same.

contact surface. If, for example, as an initial condition, two identical compliant objects are exactly superimposed, then the PFC contact surface will be missing altogether.

Missing or incomplete PFC contact surfaces do not pose a real problem. In the physical world, the probability of exactly equal gradients is 0. In simulation, exactly equal gradients will occur with finite probability because of human-set initial conditions (e.g., dropping boxes on top of each other as shown in Fig. C.1). But, even with human-set initial conditions, this issue doesn't cause simulation issues in practice. For example, I began a simulation with 6 exactly superimposed boxes resting on the ground. The result of this simulation was the boxes flying up in the air one at a time. The boxes shouldn't have interacted, but they did because of numerical error during integration. In short, a missing or incomplete PFC contact surface is not an issue in practice; even if it occurs because of human-set initial conditions, it won't be sustained in the presence of integrator error.

C.3 Computing characteristic thickness

As described in §5.2.1, I choose to give each compliant object a damping constant χ. The calculation of χ requires a characteristic thickness L. Here I explain how to calculate L. Characteristic thickness quantifies the average thickness of a compliant region. I assume that e, the extent field described in §5.1, is 0 on the surface and that $e = 1$ at rigid boundaries (see, e.g., Fig. 5.2).

The left panel of Fig. C.2 shows a rigid core bordered above and below by compliant layers with thickness h. I have partitioned each layer into triangle meshes in such a way that $e = 0$ or $e = 1$ at all triangle vertices. The characteristic thickness of each triangle L_{tri} is equal to the layer height as follows:

$$L_{tri} \equiv \text{height} = \|\nabla e\|^{-1}. \tag{C.8}$$

The characteristic thickness is the same for the left and right panels of Fig. C.2 because the average height of the compliant layers in the two panels is identical. I want to find a formula for characteristic object thickness that is a weighted average of triangle characteristic thickness. I therefore need to find weights w such that the formula:

$$L \equiv \frac{\sum L_{tri} w_j}{\sum w_j} \tag{C.9}$$

produces the same answer for the left and right panels. For triangles in this figure, I use the term 'base' to refer to the edge whose vertices have the same e. A suitable w_j for the j^{th} triangle is:

$$w_j = \int \|\nabla e\| \; dA_{\text{tri}_j} \tag{C.10}$$

$$= \frac{1}{2} \cdot \text{base}_j \cdot \underbrace{\text{height}_j}_{\|\nabla e\|^{-1}} \cdot \|\nabla e\|. \tag{C.11}$$

The reasoning in this appendix applies to triangles and tets equally. The characteristic length determined by this formula is a property of an extent field, and is invariant to meshing. For tets, the formula for calculating L is (5.8) for the weights in (5.9).

C.4 Deriving χ for compliant-compliant contact

Here I explain how to derive (5.14), the formula for χ for compliant-compliant contacts for use in (5.11). This derivation uses two expressions for $\dot{u}_d$, the rate of total energy change from normal damping. The first expression for $\dot{u}_d$ is the rate of energy change from normal damping by p_χ at Q which is:

$$\dot{u}_d = \underbrace{p_\chi \hat{\boldsymbol{n}}}_{\text{traction}} \cdot \boldsymbol{v}_{B/A} = -p_0 \chi (\boldsymbol{v}_{B/A} \cdot \hat{\boldsymbol{n}})^2. \tag{C.12}$$

The second expression for $\dot{u}_d$ is the sum of $\dot{u}_{dA}$, the rate of energy change from A, and $\dot{u}_{dB}$, the rate of energy change from B, as follows:

$$\dot{u}_d = \underbrace{-p_0 \chi_A v_{\hat{\boldsymbol{n}}_A}^2}_{\dot{u}_{dA}} + \underbrace{-p_0 \chi_B v_{\hat{\boldsymbol{n}}_B}^2}_{\dot{u}_{dB}}. \tag{C.13}$$

Next, I simplify (C.13) using the expressions for $v_{\hat{\boldsymbol{n}}_A}$ and $v_{\hat{\boldsymbol{n}}_B}$ in (C.22) and (C.23), and then rearrange terms (C.13) to obtain:

$$\dot{u}_d = -p_0 \underbrace{\left(\frac{\chi_A c_A^2 + \chi_B c_B^2}{(c_A + c_B)^2} \right)}_{\chi} (\boldsymbol{v}_{B/A} \cdot \hat{\boldsymbol{n}})^2. \tag{C.14}$$

By comparing (C.12) and (C.14), I arrive at the expression for χ in (5.14).

Writing $v_{\hat{\boldsymbol{n}}_A}$ and $v_{\hat{\boldsymbol{n}}_B}$ in terms of $\boldsymbol{v}_{B/A} \cdot \hat{\boldsymbol{n}}$. Recall that Q denotes a point (e.g., a quadrature point) on the PFC contact surface. I use $\boldsymbol{v}_{Q/A}$ to denote the

relative velocity of Q with respect to the material point in A, that is instantaneously coincident with Q as follows:

$$\boldsymbol{v}_{Q/A} \equiv \boldsymbol{v}_Q - \boldsymbol{v}_A. \tag{C.15}$$

I differentiate $0 = p_{0_B} - p_{0_A}$ at Q to obtain the following:

$$0 = \nabla p_{0_B} \cdot \boldsymbol{v}_{Q/B} - \nabla p_{0_A} \cdot \boldsymbol{v}_{Q/A}. \tag{C.16}$$

I write the velocities in (C.16) in terms of components normal and tangent to the PFC contact surface as follows:

$$\begin{aligned} 0 = \nabla p_{0_B} \cdot (\boldsymbol{v}_{Q/B} - \hat{\boldsymbol{n}}(\boldsymbol{v}_{Q/B} \cdot \hat{\boldsymbol{n}}) + \hat{\boldsymbol{n}}(\boldsymbol{v}_{Q/B} \cdot \hat{\boldsymbol{n}})) \\ -\nabla p_{0_A} \cdot (\underbrace{\boldsymbol{v}_{Q/A} - \hat{\boldsymbol{n}}(\boldsymbol{v}_{Q/A} \cdot \hat{\boldsymbol{n}})}_{\text{tangent to } \hat{\boldsymbol{n}}} + \underbrace{\hat{\boldsymbol{n}}(\boldsymbol{v}_{Q/A} \cdot \hat{\boldsymbol{n}})}_{\text{normal to } \hat{\boldsymbol{n}}}). \end{aligned} \tag{C.17}$$

In (C.17), the velocity components tangent to $\hat{\boldsymbol{n}}$ cancel. The expansion velocity of an object i at Q is as follows:

$$v_{\hat{\boldsymbol{n}}_i} \equiv \boldsymbol{v}_{Q/i} \cdot \hat{\boldsymbol{n}}_i. \tag{C.18}$$

Noting that $\hat{\boldsymbol{n}}_A = \hat{\boldsymbol{n}}$ and that $\hat{\boldsymbol{n}}_B = -\hat{\boldsymbol{n}}$, I simplify (C.17) as follows:

$$0 = \underbrace{\nabla p_{0_B} \cdot \hat{\boldsymbol{n}}}_{\approx c_B^{-1}} \underbrace{(\boldsymbol{v}_{Q/B} \cdot \hat{\boldsymbol{n}})}_{-v_{\hat{\boldsymbol{n}}_B}} - \underbrace{\nabla p_{0_A} \cdot \hat{\boldsymbol{n}}}_{\approx -c_A^{-1}} \underbrace{(\boldsymbol{v}_{Q/A} \cdot \hat{\boldsymbol{n}})}_{v_{\hat{\boldsymbol{n}}_A}} \tag{C.19}$$

$$0 = c_B^{-1} v_{\hat{\boldsymbol{n}}_B} - c_A^{-1} v_{\hat{\boldsymbol{n}}_A}. \tag{C.20}$$

The sum of the expansion velocities in objects A and B is their relative velocity projected in the $\hat{\boldsymbol{n}}$ direction as follows:

$$\boldsymbol{v}_{B/A} \cdot \hat{\boldsymbol{n}} = v_{\hat{\boldsymbol{n}}_A} + v_{\hat{\boldsymbol{n}}_B}. \tag{C.21}$$

Solving (C.21) and (C.20) for $v_{\hat{\boldsymbol{n}}_A}$ and $v_{\hat{\boldsymbol{n}}_B}$ gives:

$$v_{\hat{\boldsymbol{n}}_A} = \frac{c_A}{c_B + c_A}(\boldsymbol{v}_{B/A} \cdot \hat{\boldsymbol{n}}) \text{ and} \tag{C.22}$$

$$v_{\hat{\boldsymbol{n}}_B} = \frac{c_B}{c_A + c_B}(\boldsymbol{v}_{B/A} \cdot \hat{\boldsymbol{n}}). \tag{C.23}$$

C.5 Continuous forces from discrete geometry

Some contact models, such as those that combine minimum translational distance and point-contact, produce wrenches that are not continuous functions of state (see, e.g., [48]). The PFC implementation in §5 produces continuous wrenches when the conditions in §A.2 are satisfied, which they generally are. The contact wrench $\boldsymbol{f}$ in (4.1), is the integral of traction contributions $\boldsymbol{t}$ over $\mathcal{S}$, the configuration-dependent PFC contact surface between deformed objects A and B. A has a deformed boundary that includes the PFC contact surface, and the rest of A. Traction $\boldsymbol{t}$ is only non-zero where A and B touch (i.e., on $\mathcal{S}$). I can therefore modify (4.1) and calculate $\boldsymbol{f}$ over $\mathcal{A}$, the deformed boundary of A, instead of over $\mathcal{S}$ as follows:

$$\boldsymbol{f} = \int \begin{bmatrix} \boldsymbol{r} \times \boldsymbol{t} \\ \boldsymbol{t} \end{bmatrix} \, d\mathcal{A}. \tag{C.24}$$

The total traction $\boldsymbol{t}$ is a continuous function of $\hat{\boldsymbol{n}}$. To prove this, I combine p from (5.18) and $\boldsymbol{t}$ from (5.19) to produce the expression for $\boldsymbol{t}$ in (C.25).

$$\boldsymbol{t} = \underbrace{p_0 \max(1 + \chi(\boldsymbol{v}_{B/A} \cdot \hat{\boldsymbol{n}}), 0)}_{p} (\hat{\boldsymbol{n}} + \underbrace{\bar{\boldsymbol{t}}_f}_{\text{continuous function of } \boldsymbol{v}_T}). \tag{C.25}$$

The relative tangential velocity $\boldsymbol{v}_T$ is a continuous function of $\boldsymbol{v}_{B/A}$ and $\hat{\boldsymbol{n}}$ (see (4.9)). The relative velocity $\boldsymbol{v}_{B/A}$ is a continuous function of state. Pressure fields p_0 are immutable and spatially continuous. It follows that $\boldsymbol{t}$ is a continuous function of state if $\hat{\boldsymbol{n}}$ is a continuous function of state.

Conditions for continuous wrenches. Recall that $\mathcal{A}$ is the deformed boundary of A. When $\mathcal{A}$ moves continuously, so does $\hat{\boldsymbol{n}}$, and therefore so do tractions. It follows that PFC produces wrenches that are a continuous function of state if $\mathcal{A}$ is

a continuous function of state. For rigid-compliant contacts, if one calls the rigid object A, the boundary of A (i.e., $\mathcal{A}$) is a continuous function of state, so PFC produces continuous wrenches. For compliant-compliant contacts, $\mathcal{A}$ is a continuous function of state when the pressure field gradients are oriented in opposite directions (i.e., $\nabla p_{0_A} \cdot \nabla p_{0_B} \leq 0$) at all points on $\mathcal{S}$.

APPENDIX D

TANGENTIAL COMPLIANCE

D.1 Deriving the spatial tangential compliance matrix

I show that $\boldsymbol{f}_{\boldsymbol{\Delta}}$, the PFC contact surface friction wrench generated by springs, can be written in the form shown in (6.11) and repeated below.

$$\boldsymbol{f}_{\boldsymbol{\Delta}} = -\boldsymbol{K}\boldsymbol{\Delta},$$

$$\text{where} \quad \boldsymbol{\Delta} \equiv \begin{bmatrix} \boldsymbol{\Delta}_{\theta} \\ \boldsymbol{\Delta}_{\boldsymbol{r}} \end{bmatrix}$$

The traction applied to B from the spring is $-k\boldsymbol{\delta}$. Integrating spring traction contributions over the entire PFC contact surface gives

$$\boldsymbol{f}_{\boldsymbol{\Delta}} = -\int k \begin{bmatrix} \boldsymbol{r} \times \boldsymbol{\delta} \\ \boldsymbol{\delta} \end{bmatrix} \mathcal{S} \tag{D.1}$$

This derivation requires writing three mathematical expressions as matrix operations.

1. $\perp \hat{\boldsymbol{n}}$ **as a matrix multiply:** For an arbitrary vector $\boldsymbol{v}$, the operation $\boldsymbol{v} \perp \hat{\boldsymbol{n}}$ as defined in (6.10) can be written as $\hat{\boldsymbol{n}}_{\perp}\boldsymbol{v}$ for the matrix $\hat{\boldsymbol{n}}_{\perp}$ shown below:

$$\hat{\boldsymbol{n}}_{\perp} \equiv I_3 - \hat{\boldsymbol{n}}\hat{\boldsymbol{n}}^T \tag{D.2}$$

2. $\boldsymbol{r}\times$ **as a matrix:** For an arbitrary vector $\boldsymbol{v}$, the operation $\boldsymbol{r} \times \boldsymbol{v}$ can be written as $\boldsymbol{r}_{\times}\boldsymbol{v}$, where $\boldsymbol{r}_{\times}$ is

$$\boldsymbol{r}_{\times} \equiv \begin{bmatrix} 0 & -\boldsymbol{r}_3 & \boldsymbol{r}_2 \\ \boldsymbol{r}_3 & 0 & -\boldsymbol{r}_1 \\ -\boldsymbol{r}_2 & \boldsymbol{r}_1 & 0 \end{bmatrix} \tag{D.3}$$

3. **$\boldsymbol{\Delta}_\theta \times \boldsymbol{r} + \boldsymbol{\Delta}_{\boldsymbol{r}}$ as a matrix multiply:** Using (D.3), the quantity $\boldsymbol{\Delta}_\theta \times \boldsymbol{r} + \boldsymbol{\Delta}_{\boldsymbol{r}}$ be written as

$$\boldsymbol{\Delta}_\theta \times \boldsymbol{r} + \boldsymbol{\Delta}_{\boldsymbol{r}} = \begin{bmatrix} -\boldsymbol{r}_\times & I_3 \end{bmatrix} \boldsymbol{\Delta} \tag{D.4}$$

Using the definition of $\boldsymbol{\delta}$ from (6.10) and the matrix relationships in (D.2) and (D.4) I write $\boldsymbol{\delta}$ as follows:

$$\boldsymbol{\delta} = \hat{\boldsymbol{n}}_\perp \begin{bmatrix} -\boldsymbol{r}_\times & I_3 \end{bmatrix} \boldsymbol{\Delta} \tag{D.5}$$

I plug $\boldsymbol{\delta}$ from (D.5) into (D.1) to get

$$\boldsymbol{f}_{\boldsymbol{\Delta}} = -\underbrace{\int k \begin{bmatrix} -\boldsymbol{r}_\times \hat{\boldsymbol{n}}_\perp \boldsymbol{r}_\times & \boldsymbol{r}_\times \hat{\boldsymbol{n}}_\perp \\ -\hat{\boldsymbol{n}}_\perp \boldsymbol{r}_\times & \hat{\boldsymbol{n}}_\perp \end{bmatrix} d\mathcal{S}}_{\boldsymbol{K}} \boldsymbol{\Delta}. \tag{D.6}$$

D.2 Dimension reduction

The friction wrench $\boldsymbol{f}_c$ is the sum of a wrench from springs and a wrench from dampers. To calculate the damper wrench $\boldsymbol{f}_{\dot{\boldsymbol{\Delta}}|_{\boldsymbol{K}}}$, I differentiate $\boldsymbol{\delta}$ of the constant stiffness spring to get

$$\dot{\boldsymbol{\delta}}|_k^C = (\dot{\boldsymbol{\Delta}}|_{\boldsymbol{K}\theta}^C \times \boldsymbol{r}^C + \dot{\boldsymbol{\Delta}}|_{\boldsymbol{K}r}^C) \perp \hat{\boldsymbol{n}}^C, \tag{D.7}$$

and use the approach used to get $\boldsymbol{f}_{\boldsymbol{\Delta}}$ in §D.1 to get

$$\boldsymbol{f}_{\dot{\boldsymbol{\Delta}}|_{\boldsymbol{K}}}^C = -\tau \boldsymbol{K}^C \dot{\boldsymbol{\Delta}}|_{\boldsymbol{K}}^C. \tag{D.8}$$

Adding $\boldsymbol{f}_{\dot{\boldsymbol{\Delta}}|_{\boldsymbol{K}}}$ from (D.8) to $\boldsymbol{f}_{\boldsymbol{\Delta}}$ gives the total wrench

$$\boldsymbol{f}_c^C = \boldsymbol{f}_{\boldsymbol{\Delta}} - \tau \boldsymbol{K}^C \dot{\boldsymbol{\Delta}}|_{\boldsymbol{K}}^C. \tag{D.9}$$

Rearranging (D.9) produces (6.13).

BIBLIOGRAPHY

[1] Jérémie Allard, François Faure, Hadrien Courtecuisse, Florent Falipou, Christian Duriez, and Paul G Kry. Volume contact constraints at arbitrary resolution. In *ACM SIGGRAPH 2010 papers*, pages 1–10. 2010.

[2] Tobias FC Berninger, Felix Sygulla, Sebastian Fuderer, and Daniel J Rixen. Experimental analysis of structural vibration problems of a biped walking robot. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8726–8731. IEEE, 2020.

[3] Antonio Bicchi and Vijay Kumar. Robotic grasping and contact: A review. In *Proc. IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 348–353, San Francisco, CA, April 2000.

[4] C. Bouchard, M. Nesme, M. Tournier, B. Wang, F. Faure, and P. G. Kry. 6d frictional contact for rigid bodies. In *Proceedings of Graphics Interface 2015*, GI '15. Canadian Human Computer Interaction Society, 2015.

[5] Justin Carpentier, Guilhem Saurel, Gabriele Buondonno, Joseph Mirabel, Florent Lamiraux, Olivier Stasse, and Nicolas Mansard. The pinocchio c++ library: A fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives. In *2019 IEEE/SICE International Symposium on System Integration (SII)*, pages 614–619. IEEE, 2019.

[6] Erin Catto. Iterative dynamics with temporal coherence. In *Game developer conference*, volume 2, 2005.

[7] Anindya Chatterjee. Asymptotic solution for solitary waves in a chain of elastic spheres. *Phys. Rev. E*, 59:5912–5919, May 1999.

[8] Minxin Chen, Bin Tu, and Benzhuo Lu. Surface triangular mesh and volume tetrahedral mesh generations for biomolecular modeling. In *Image-Based Geometric Modeling and Mesh Generation*, pages 85–106. Springer, 2013.

[9] Se-Joon Chung and Nancy Pollard. Predictable behavior during contact simulation: a comparison of selected physics engines. *Computer Animation and Virtual Worlds*, 27(3-4):262–270, 2016.

[10] Matei Ciocarlie, Claire Lackner, and Peter Allen. Soft finger model with adaptive contact geometry for grasping and manipulation tasks. In *Second*

Joint EuroHaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems (WHC'07), pages 219–224. IEEE, 2007.

[11] Michael Danielczuk, Jingyi Xu, Jeffrey Mahler, Matthew Matl, Nuttapong Chentanez, and Ken Goldberg. Reach: Reducing false negatives in robot grasp planning with a robust efficient area contact hypothesis model. *Int. S. Robotics Research (ISRR)*, 2019.

[12] Ryan Elandt, Evan Drumwright, Michael Sherman, and Andy Ruina. A pressure field model for fast, robust approximation of net contact force and moment between nominally rigid objects. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019.

[13] Christer Ericson. *Real-time collision detection*. Crc Press, 2004.

[14] Kenny Erleben. Methodology for assessing mesh-based contact point methods. *ACM Trans. on Graphics*, 37(3), 2018.

[15] Francois Faure. Volume based contact. 2014.

[16] François Faure, Jérémie Allard, Florent Falipou, and Sébastien Barbier. Image-based collision detection and response between arbitrary volumetric objects. In *ACM Siggraph/Eurographics Symposium on Computer Animation*, pages 155–162. Eurographics Association, 2008.

[17] Roy Featherstone. *Rigid Body Dynamics Algorithms*. Springer, 2008.

[18] Benjamin J Fregly, Yanhong Bei, and Mark E Sylvester. Experimental evaluation of an elastic foundation model to predict contact pressures in knee replacements. *Journal of biomechanics*, 36(11):1659–1668, 2003.

[19] Zhi-Fang Fu and Jimin He. *Modal analysis*. Elsevier, 2001.

[20] Markus Giftthaler, Michael Neunert, Markus Stäuble, Marco Frigerio, Claudio Semini, and Jonas Buchli. Automatic differentiation of rigid body dynamics for optimal control and estimation. *Advanced Robotics*, 31(22):1225–1237, 2017.

[21] Yves Gonthier. Contact dynamics modelling for robotic task simulation. 2007.

[22] Suresh Goyal, Andy Ruina, and Jim Papadopoulos. Planar sliding with dry friction part 1. limit surface and moment function. *Wear*, 143(2):307–330, 1991.

[23] Ernst Hairer and Gerhard Wanner. Stiff differential equations solved by radau methods. *Journal of Computational and Applied Mathematics*, 111(1-2):93–111, 1999.

[24] Shoichi Hasegawa and Nobuaki Fujii. Real-time rigid body simulation based on volumetric penalty method. In *Proc. of the 11th Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems (HAPTICS)*, 2003.

[25] Kris Hauser. Robust contact generation for robot simulation with unstructured meshes. In *Proc. Intl. Symp. Robotics Research (ISRR)*, 2013.

[26] Nicolas Heess, Dhruva TB, Srinivasan Sriram, Jay Lemmon, Josh Merel, Greg Wayne, Yuval Tassa, Tom Erez, Ziyu Wang, SM Eslami, et al. Emergence of locomotion behaviours in rich environments. *arXiv preprint arXiv:1707.02286*, 2017.

[27] Abhinandan Jain. *Robot and multibody dynamics: analysis and algorithms*. Springer Science & Business Media, 2010.

[28] Tobias Johannink, Shikhar Bahl, Ashvin Nair, Jianlan Luo, Avinash Kumar, Matthias Loskyll, Juan Aparicio Ojea, Eugen Solowjow, and Sergey Levine. Residual reinforcement learning for robot control. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6023–6029. IEEE, 2019.

[29] Kenneth L. Johnson. *Contact Mechanics*. Cambridge Univ. Press, 1987.

[30] Benjamin S Kirk, John W Peterson, Roy H Stogner, and Graham F Carey. libmesh: a c++ library for parallel adaptive mesh refinement/coarsening simulations. *Engineering with Computers*, 22(3):237–254, 2006.

[31] Twan Koolen and contributors. Rigidbodydynamics.jl, 2016.

[32] Vladimir Ivanovich Krylov and Arthur H Stroud. *Approximate calculation of integrals*. Courier Corporation, 2006.

[33] Jianmin Long, Yue Ding, Weike Yuan, Wen Chen, and Gangfeng Wang. Gen-

eral relations of indentations on solids with surface tension. *Journal of Applied Mechanics*, 84(5):051007, 2017.

[34] Lianzhen Luo and Meyer Nahon. A compliant contact model including interference geometry for polyhedral objects. *Trans. ASME*, Apr 2006.

[35] David Millard, Eric Heiden, Shubham Agrawal, and Gaurav S Sukhatme. Automatic differentiation and continuous sensitivity analysis of rigid body dynamics. *arXiv preprint arXiv:2001.08539*, 2020.

[36] Brian Mirtich. Efficient algorithms for two-phase collision detection. *Practical motion planning in robotics: current approaches and future directions*, pages 203–223, 1997.

[37] SE Mousavi, H Xiao, and N26543611183 Sukumar. Generalized gaussian quadrature rules on arbitrary polygons. *International Journal for Numerical Methods in Engineering*, 82(1):99–113, 2010.

[38] An Nguyen. Implicit bounding volumes and bounding volume hierarchies. *Doctor of Philosophy, Stanford University*, 2006.

[39] Brian Plancher, Sabrina M Neuman, Radhika Ghosal, Scott Kuindersma, and Vijay Janapa Reddi. Grid: Gpu-accelerated rigid body dynamics with analytical gradients. *arXiv preprint arXiv:2109.06976*, 2021.

[40] William Roshan Quadros, Steven J Owen, Michael L Brewer, and Kenji Shimada. Finite element mesh sizing for surfaces using skeleton. In *IMR*, pages 389–400, 2004.

[41] Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*, 2017.

[42] Aravind Rajeswaran, Kendall Lowrey, Emanuel V Todorov, and Sham M Kakade. Towards generalization and simplicity in continuous control. *Advances in Neural Information Processing Systems*, 30, 2017.

[43] Michael A Sherman, Ajay Seth, and Scott L Delp. Simbody: multibody dynamics for biomedical research. *Procedia Iutam*, 2:241–261, 2011.

[44] Russell Smith et al. Open dynamics engine, 2007.

[45] G Lakshmi Srinivas and Arshad Javed. Optimization approaches of industrial serial manipulators to improve energy efficiency: a review. In *IOP Conference Series: Materials Science and Engineering*, volume 912, page 032058. IOP Publishing, 2020.

[46] Yuval Tassa, Tom Erez, and Emanuel Todorov. Synthesis and stabilization of complex behaviors through online trajectory optimization. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4906–4913. IEEE, 2012.

[47] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE, 2012.

[48] Jedediyah Williams, Ying Lu, and Jeff C. Trinkle. A complementarity based contact model for geometrically accurate treatment of polytopes in simulation. In *Proc. ASME Intl. Design Engr. Tech. Conf. and Comput. and Inform. in Engr. Conf.*, 2014.